

# THE ULTRASWITCH

- a remoteqth project by DM5XX -



# remoteQTH.com

- OPEN-SOURCE Hardware + OPEN-SOURCE Firmware
- OK1HRA, Dan | OK2ZAW, Jan | OK1CDJ, Ondra | DM5XX, Mike
- <https://remoteqth.com> + [qro.cz](http://qro.cz) + [hamshop.cz](http://hamshop.cz) = [OL7M](#)
- K9AY
- Beverages: Top band, single band, bi-directional 2-wire, 4-way bi-directional,...
- Switching:  
6To2/6To1/4To1, 6To2+Triplexer, 1To2/1To3 Stackmatch, 12x4,...
- Variable ATT, SO2R, M/M, Controller (on-site/remote), Custom Contest Hardware...



# DM5XX

- Funkamateur seit 1990 (OK8XX, DM5X, ex. DG3NED, ex. DL3NED) aus Nürnberg
- Senior Software Engineer
- Derzeit IT-Projektleiter (Leitung Niederlassung) @ ADITUS GmbH
- Seit 2015 Lead Firmware Developer @remoteqth.com
- Hardwarenahe Microcontrollerprogrammierung



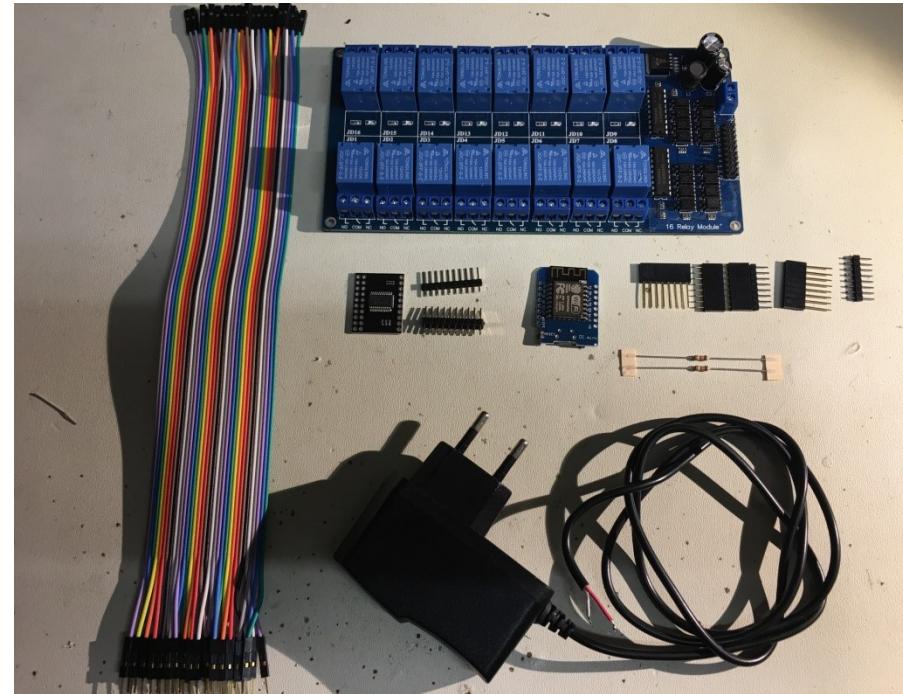
# The Ultraswitch

- Wifi basiert & inspiriert vom [mega-webswitch](#) (63-CH remote-switch)
- Teil des 256-Port-Inband-Antennenprojekts @ OL7M
- Einfache Hardware direkt aus China (see ebay.com :P)
- ESP8266-12F („Wemos D1 mini“)
- Modular: Bis zu 8 Boards
- Web-UI & Web-Config
- RESTful und einfach erweiterbar dank web-basierter Architektur
- Fertige und programmierte Kits => Email: dm5xx@gmx.de



# Die Hardwarekomponenten

- 16-CH Relayboard mit 12V-Spannungsversorgung
- MCP23017SMD auf PCB (I2C=>16 Port)
- Wemos D1 mini (ESP8266-12F, Controller)
- 2x 1.8K
- DuPont Steckverbinder
- 12V-Netzteil
- Computer+MicroUsb-Kabel (Progr.)



# Die Architektur

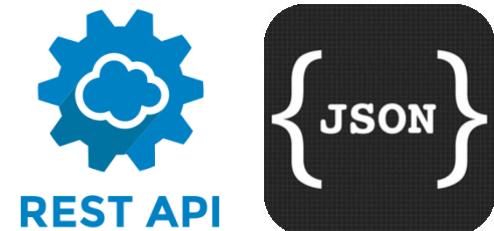
Separation of Concerns:

- Hardwareschicht (1)
- Hardwarenahe Funktionalität (JSON) (1)
- Funktionensschicht (JS) (2)
- Anzeigeschicht (UI) (3)

=> aka 3-Layer-Modell

Browser:  
(3) Anzeige

Javascript:  
(2) Steuerfunktionen



(1) ESP12F/RelayBoard



# Vor- und Nachteile

---

## Vorteile:

- Leicht erweiterbar, Unabhängig von der UI  
(Web-App, Windows-App, Mobile-App, Linux-App,...)
- Trennung von Hardware, Logik und Funktionen
- Zentralisiert, Modular, Webtechnologie, Zukunftsfähig
- Eigene Funktionen und UI (Customizing)

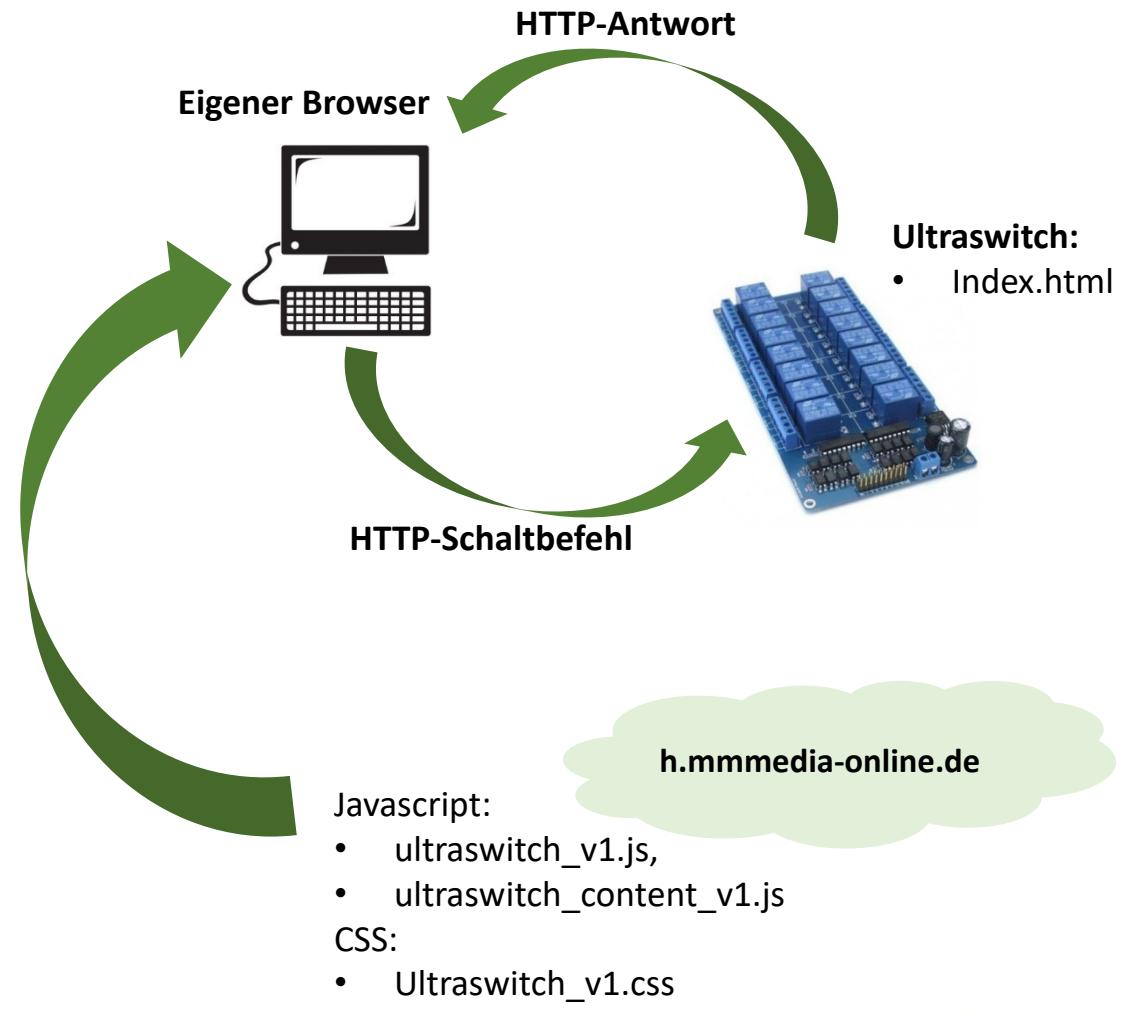
## Nachteile:

- Sicherheitsmechanismen liegen in eigener Verantwortung der Funktionsschicht
- „Viel Feind, viel Ehr“ (Je mehr angepasst wird, umso mehr Wissen ist notwendig)



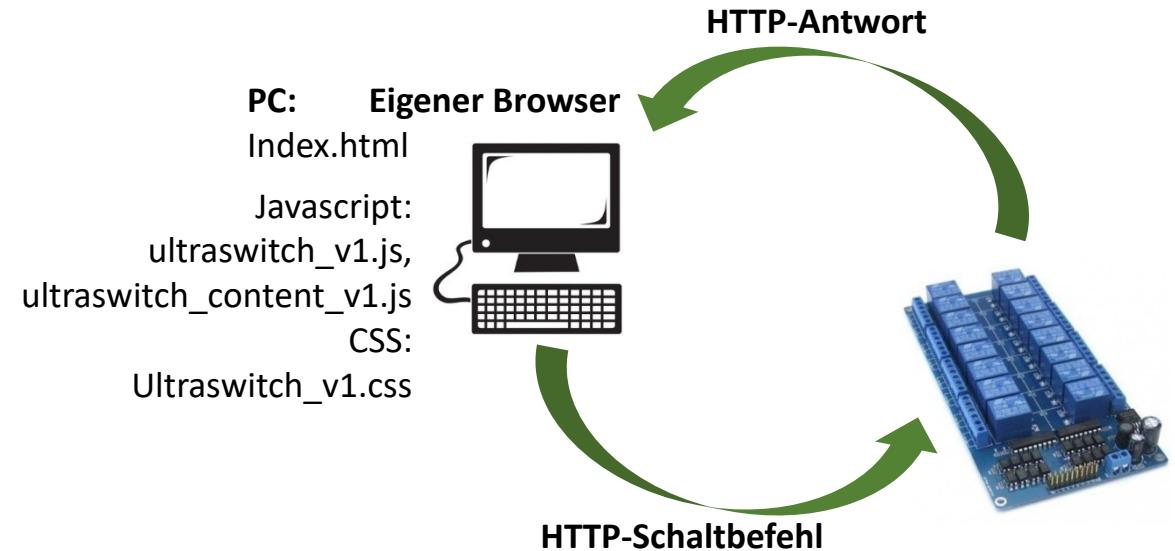
# Usecase 1 – „Out of the box“

- Keine Anpassungen notwendig
- Alle vorbereiteten Ressourcen („h.mmmmedia-online.de“) werden einfach verwendet und genutzt
- Nur Konfiguration des Switches notwendig (IP-Adresse vergeben)
- Version immer up-to-Date
- Nachteil: Beschriftung kann nicht geändert werden.



# Usecase 2 – „Eigene Desktop-Anwendung“

- Anpassungen einfach möglich
- Alle Ressourcen lokal auf dem PC
- Alles anpassbar
- Kein eigener Webserver notwendig
- Nachteil: Keine automatischen Updates, nicht mehr Multi-User-Fähig (Logik/Infos liegen auf dem eigenen Rechner).  
Nicht mobile-fähig.

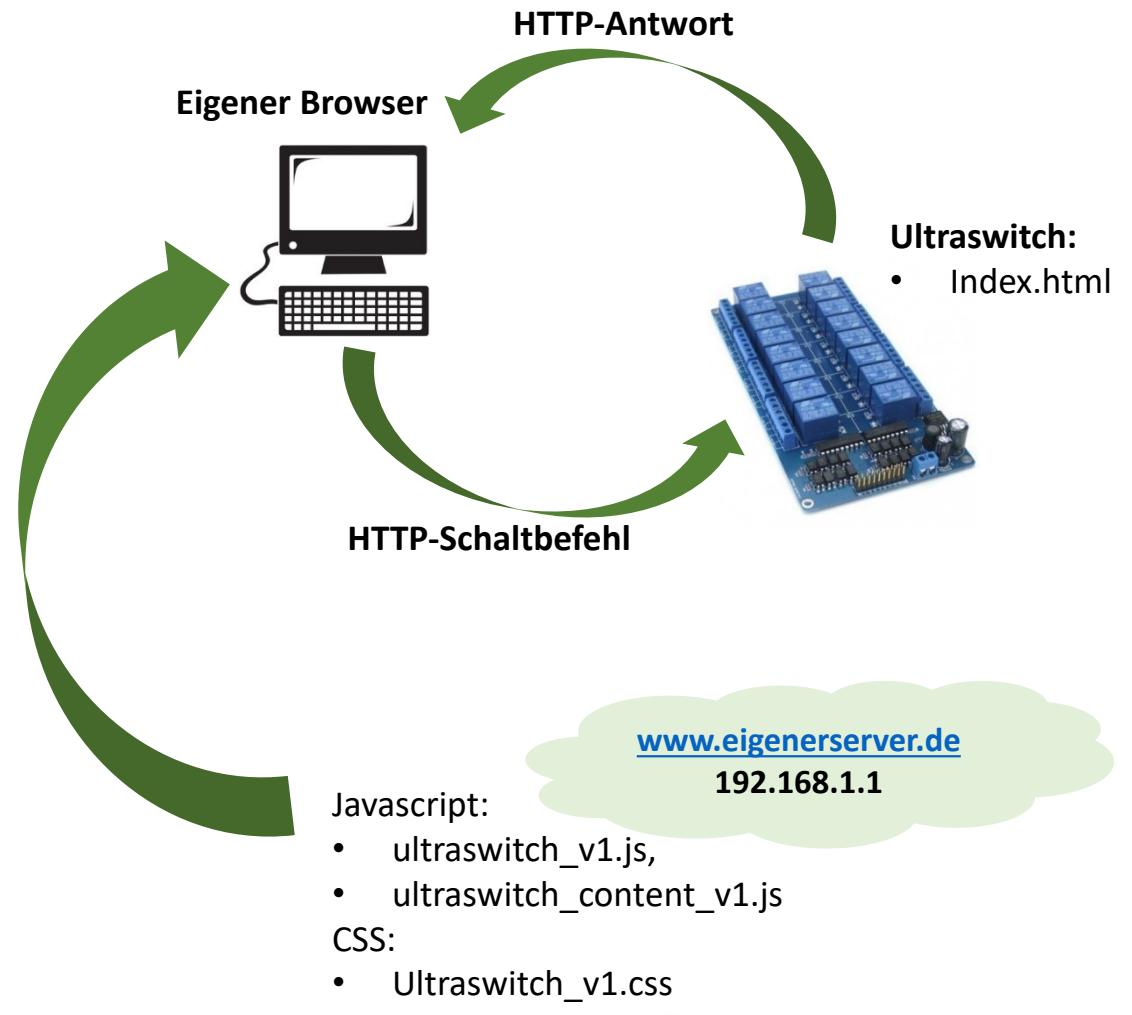


# Usecase 3 – „Eigener Webserver“

- Vereint Vorteile aus 1 uns 2
- Multi-User-Fähig
- Flexibler Zugriff
- Server kann im eigenen Netz oder extern Sein

Nachteil:

- Webspace notwendig
- WebServer-Kenntnisse notwendig
- Ebenfalls Keine automat. Updates



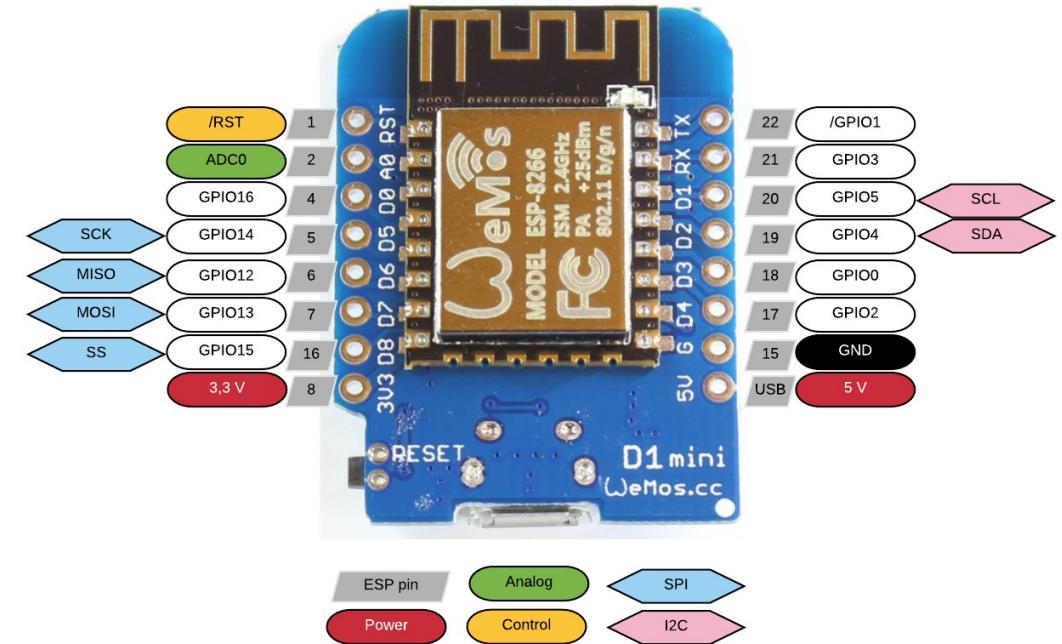
# Good to know

- Einfache Erklärung „[Netzwerkgrundlagen](#)“
- Einführung in „[Übertragungsprotokolle](#)“
- [JSON](#)
- [HTML, CSS, Javascript](#)
- Introduction into [REST](#)



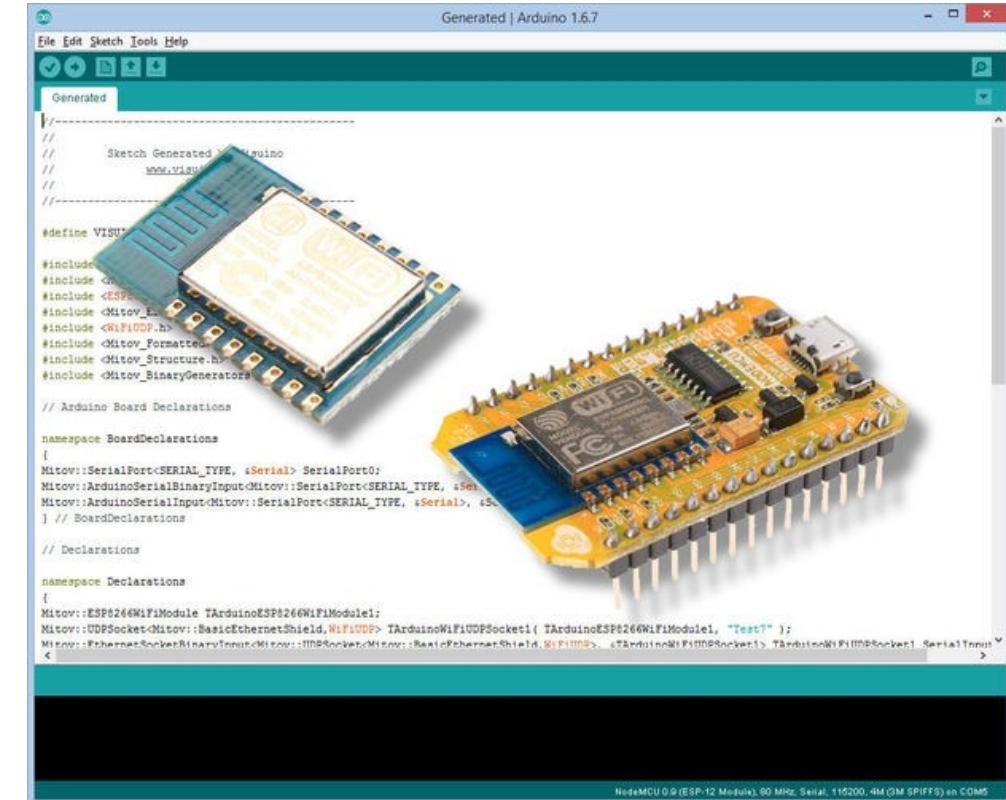
# Die Hardwareplattform – ESP8266-12F

- 802.11 b/g/n, Wi-Fi 2.4 GHz, support WPA/WPA2
- Integrated low power 32-bit MCU @80mhz, 4MB Ram, 92kb
- Integrated 10bit ADC
- Integrated TCP/IP protocol stack
- Integrated TR switch, balun, LNA,
- power amplifier and matching network
- Integrated PLL, regulators, and power management units
- Supports antenna diversity
- Support STA/AP/STA+AP operation modes
- Support Smart Link Function for both Android and iOS devices
- SDIO 2.0, (H) SPI, UART, I2C, I2S, IRDA, PWM, GPIO, STBC, 1x1 MIMO, 2x1 MIMO
- Deep sleep power <10uA, Power down leakage current < 5uA
- Standby power consumption of < 1.0mW
- <http://stefanfrings.de/esp8266/>
- [Datasheet](#)



# Arduino-IDE

- <http://www.gunook.com/einrichten-der-arduino-ide-zum-programmieren-von-esp8266/>
- [http://arduino.esp8266.com/versions/2.3.0/package\\_esp8266com\\_index.json](http://arduino.esp8266.com/versions/2.3.0/package_esp8266com_index.json)  
(Achtung: 2.3.0! 2.4.0 hat einen Bug)



# Kurzer Ausflug in die IDE und Programmierung

- <https://www.youtube.com/watch?v=0wAY3DYihyg>
- <https://www.youtube.com/watch?v=BKm1QFZS0m4>
- <https://www.youtube.com/watch?v=eaFvQG8wrGw&list=PLAB63281B90FB376E&index=3>
- <https://www.youtube.com/watch?v=t-lRfOalxhE&index=4&list=PLAB63281B90FB376E>
- <https://www.youtube.com/watch?v=nf8Uqy6vLmg&index=5&list=PLAB63281B90FB376E>
- <https://www.youtube.com/watch?v=QIDH2SA pocM&list=PLAB63281B90FB376E&index=6>
- Siehe Verzeichnis „Software“



# Libraries (Bibliotheken und der Sketch)

- [Artikel zu Libraries](#) (heise.de)““

**Nach der IDE-Installation liegen die Projektfiles...**

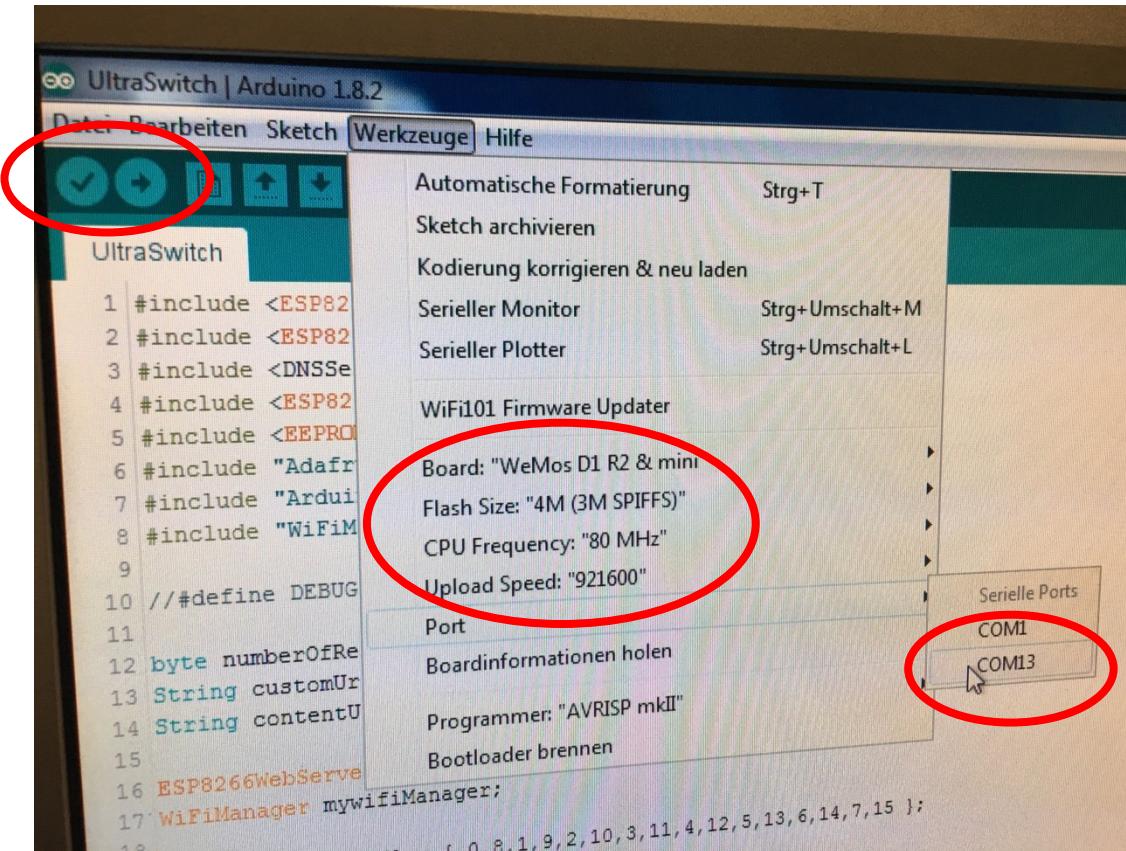
- C:\Users\mmmedia\Documents\Arduino  
=> Folder Ultraswitch dorthin kopieren
- C:\Users\mmmedia\Documents\Arduino\libraries  
=> Folder aus “Ultraswitch/needed\_libraries” hinkopieren
- Auf den Desktop kopieren: (für später)  
C:\Users\mmmedia\Documents\Arduino\Arrest\ultraSwitch\_web



# Los geht's: 8266-12F

- Board ESP hinzugefügt?
- Libraries kopiert?
- Sketch geladen?
- Einstellungen angepasst?
- Serieller Port stimmt?
- 12F mit USB verbunden?

JA? => Verify > Hochladen!

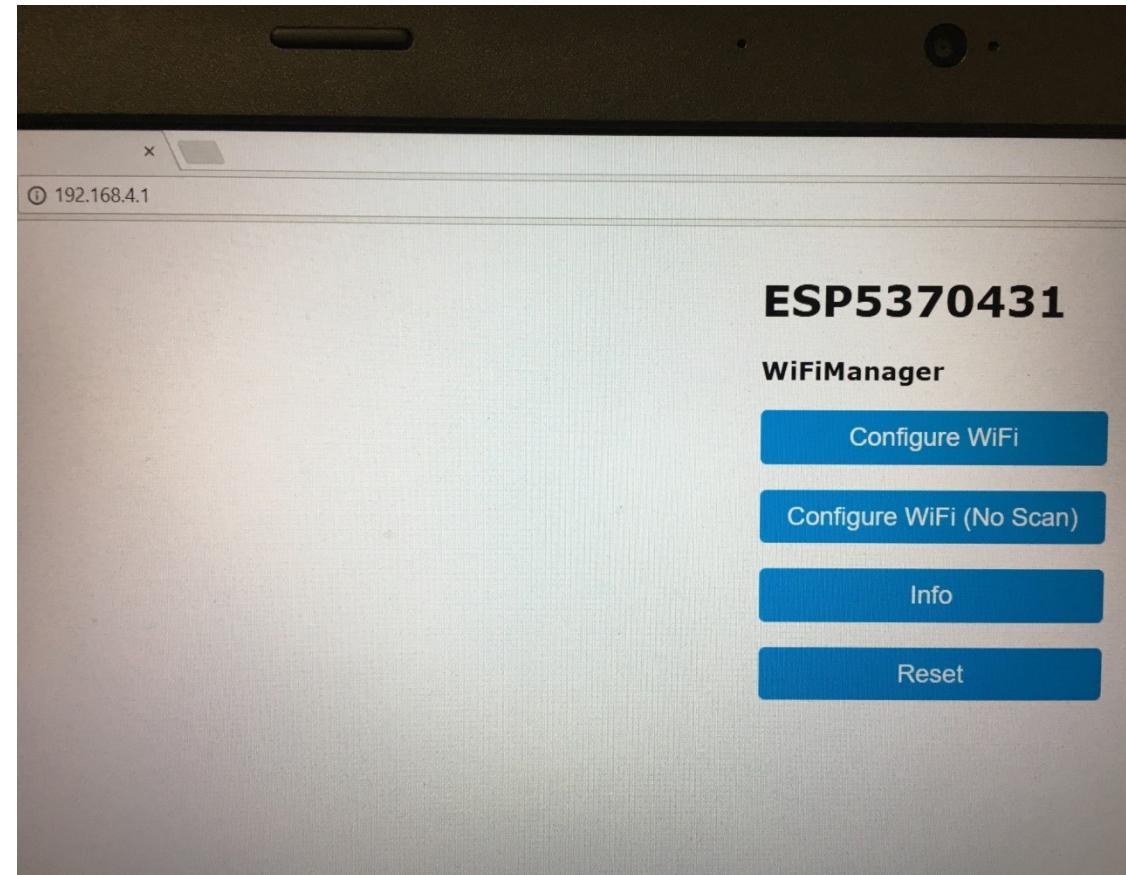


# Mit dem 12F verbinden – AP-Mode 1

- Wifi-Netz ESP<Nummer> suchen
- Mit ESP<Nummer> verbinden
- Browser öffnen (Default: 192.168.4.1) – nichts weiter tun!

## Achtung RRDXA-Workshop:

Bitte immer nur 1 ESP angesteckt haben! Sonst gibt es zu ESP-Wlans...  
Aber welcher ist Eurer...?!



# Mit dem 12F verbinden – AP-Mode 2

- Bitte setup\_esp.html aus dem Ordner „Ultraswitch\_Web“ mit Doppelklick öffnen.
- Bitte Daten gemäß Eurer Info eintragen
- Auf „Submit“ klicken

The screenshot shows a web browser window titled "CONFIG UltraSwitch". The address bar displays "file:///C:/Users/mmmedia/Desktop/ultraSwitch\_web/setup\_esp.html". The main content area is titled "Welcome to the quick n dirty configurator for the ESP\_Accesspoint!". It contains five input fields labeled (0) through (4), each with a placeholder and a value. Below these is a fifth field (5) with a placeholder and a value. At the bottom, there is a warning message and a "Submit" button.

(0) Current IP-Adress of the AP e.g. 192.168.4.1 (no spaces, "-"-separated) --->

(1) IP-Adress of the Controller e.g. 192.168.1.111 (no spaces, "-"-separated) --->

(2) IP-Adress of your Internet-Gateway(router) e.g. 192.168.1.44 (no spaces, "-"-separated) --->

(3) IP-Adress SubNetMask 255.255.255.0 --->

(4) SSID --->

(5) WLANPWD

BE CAREFUL WHAT YOU DO!! DOUBLECHECK TWICE BEFORE CLICKING SUBMIT :P

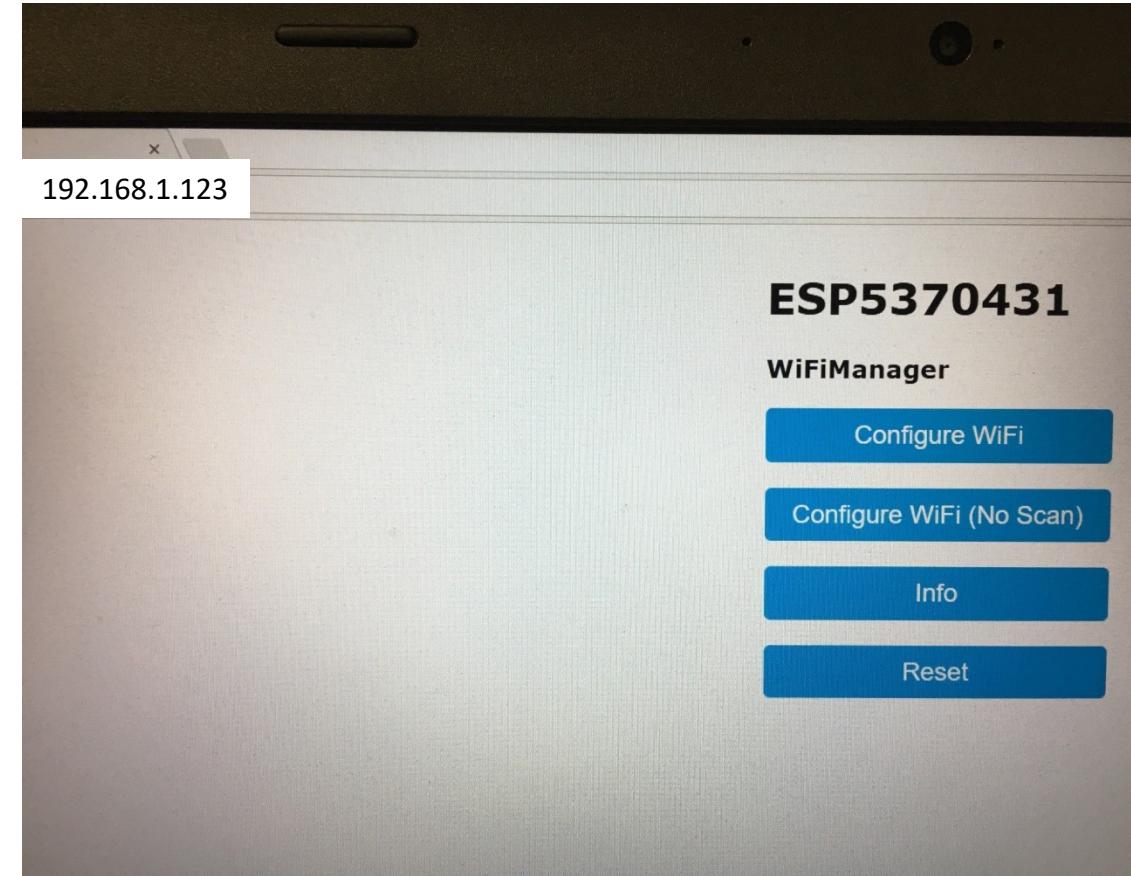


# Mit dem 12F verbinden – AP-Mode 3

- Ins richtige WLan wechseln – nicht mehr ins ESP-Wlan! (jetzt wird's tricky)
- Browser öffnen (z.B. 192.168.1.123)
- „Configure Wifi“

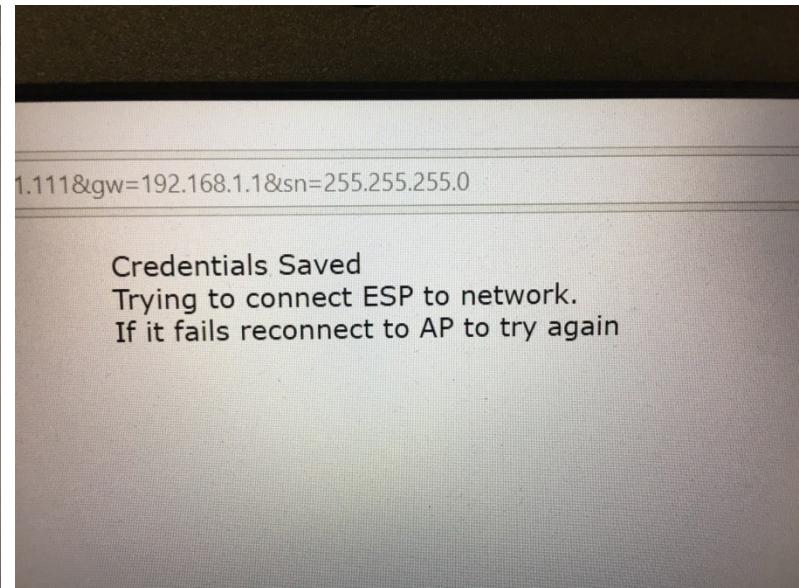
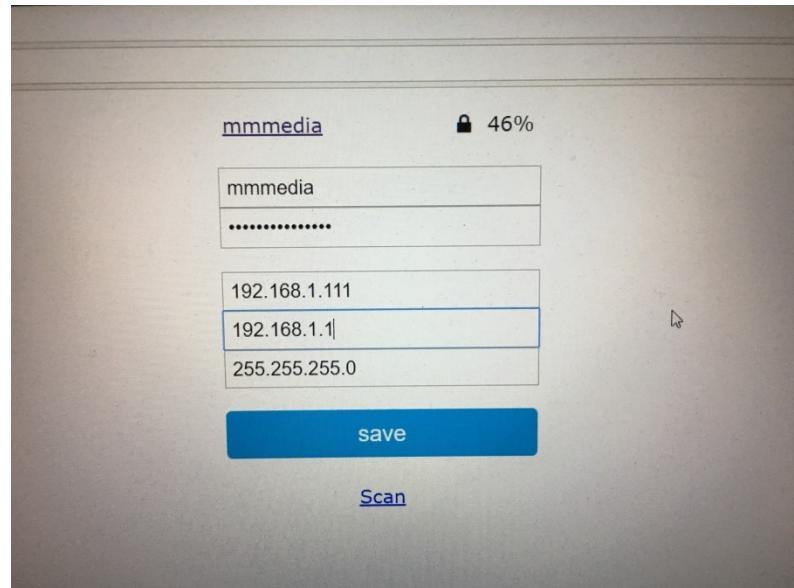
## Achtung RRDXA-Workshop:

192.168.1.123 ist die zuvor vergebene IP-Adresse, unter der Euer Ultraswitch erreichbar ist.



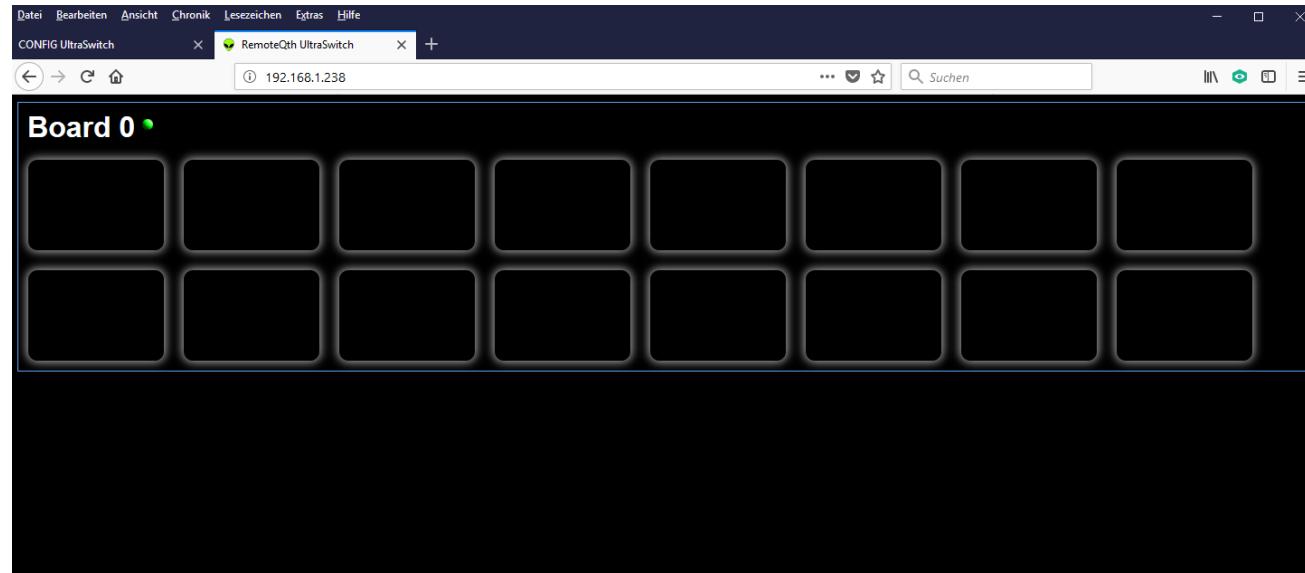
# Mit dem 12F verbinden – AP-Mode 4

- „Configure Wifi“:  
Daten SSID und PWD  
nochmals eingeben!
- Daten unten prüfen!



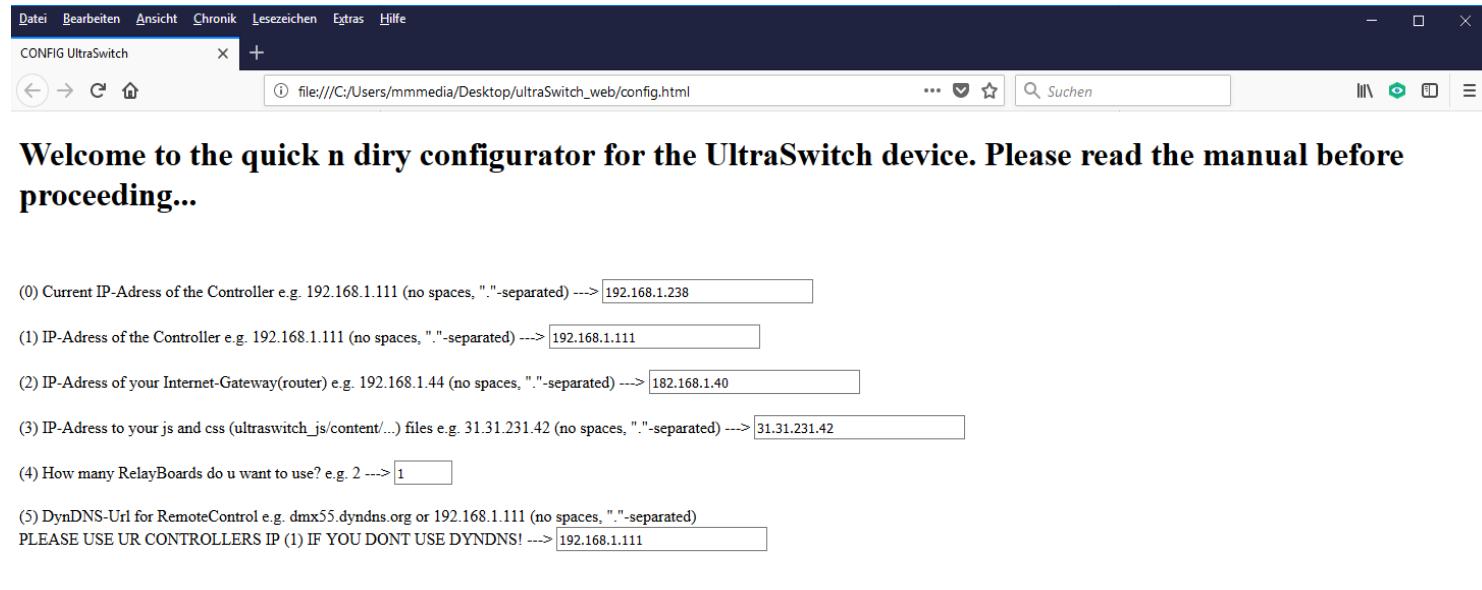
# Mit dem 12F verbinden – Der erste Kontakt

- Die IP Eures Ultraswitch im Browser aufrufen...
- Troubleshooting: Ultraswitch nicht gefunden?  
[Advanced IP-Scanner](#) hilft!  
IP merken und im Browser eingeben!
- Aber Pfade stimmen noch nicht  
=> Keine Buttons, etc...

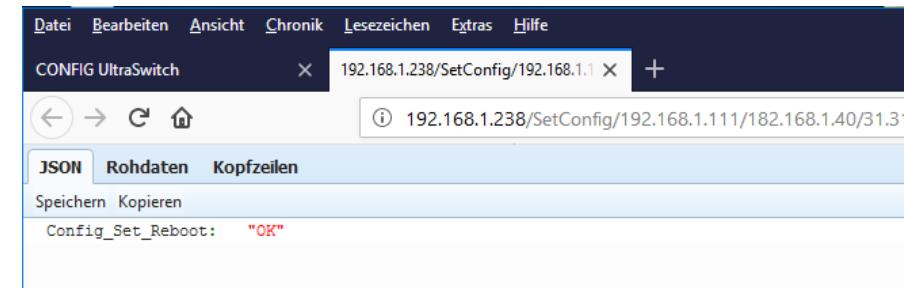


# Mit dem 12F verbinden – IP-Settings Servermode

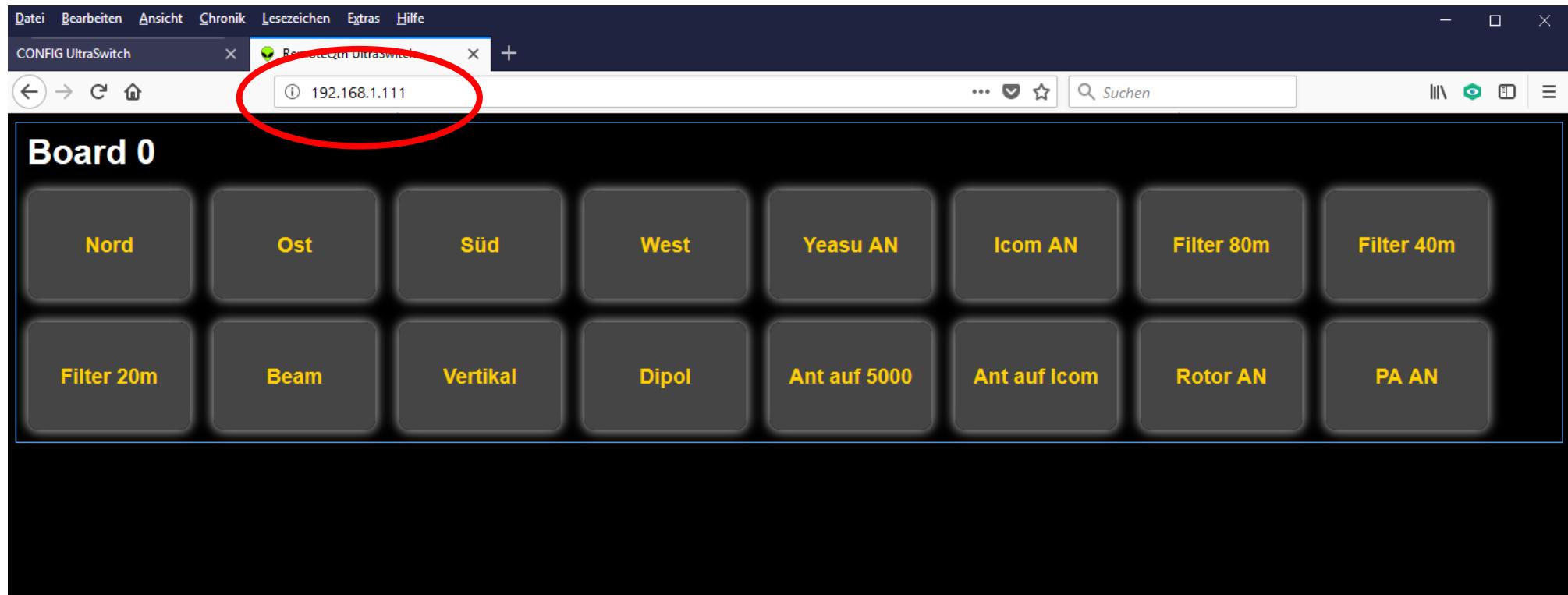
- Config.html im Verzeichnis aufrufen
- Wichtig: Derzeitige IP des Ultraswitch!
- Wichtig: Gewünscht IP des Ultraswitch!
- Restliche Information
- Submit!



BE CAREFUL WHAT YOU DO!! DOUBLECHECK TWICE BEFORE CLICKING SUBMIT :P



# Finale IP-Adresse des Ultraswitch: Die UI



(((( ))))



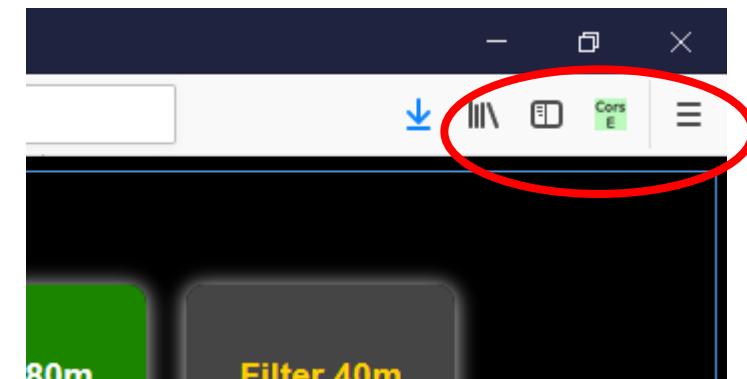
# Irgendwas geht nicht... DEBUG!

- In der ultraswitch.ino folgende Zeile (ca. Zeile 10) suchen und ändern:  
**//#define DEBUG** in **#define DEBUG** („//“ entfernen)
- Sketch neu hochladen
- Werkzeuge => Serieller Monitor
- Ggf. Übertragungsrate von 9600 auf 115200 umstellen
- Reset am Wemos (kl. Schalter am Board)
- Serieller Monitor zeigt nun IP-Adresse und Statuszustände an...
- Hinweis: Nach erfolgreicher Fehlersuche wieder //#define DEBUG



# CORS – Cross Origin Request

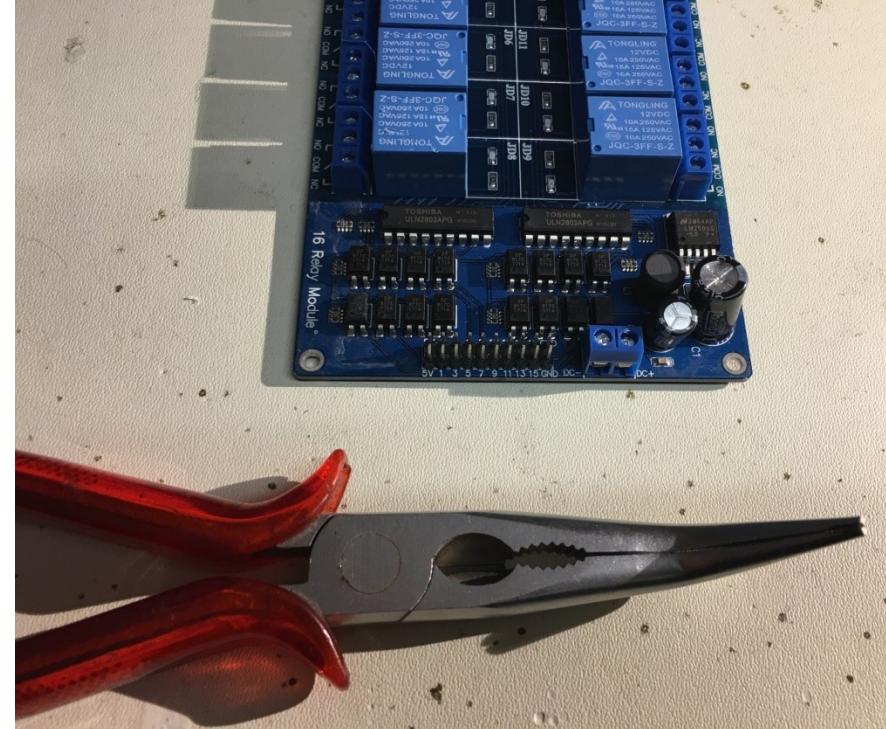
- Durch die AJAX-Requests kann es oft dazu kommen, dass die Requests durch die BrowserSicherheit blockiert werden, z.B. Beim Start der index.html vom Desktop  
(file:///C:/Users/mmmedia/../../index.html)
- Abhilfe: [CORS Everywhere](#) (Firefox)  
=> Zu Firefox hinzufügen  
=> CORS E aktivieren („CorsE“ wird Grün)
- Empfohlener Browser: Firefox!



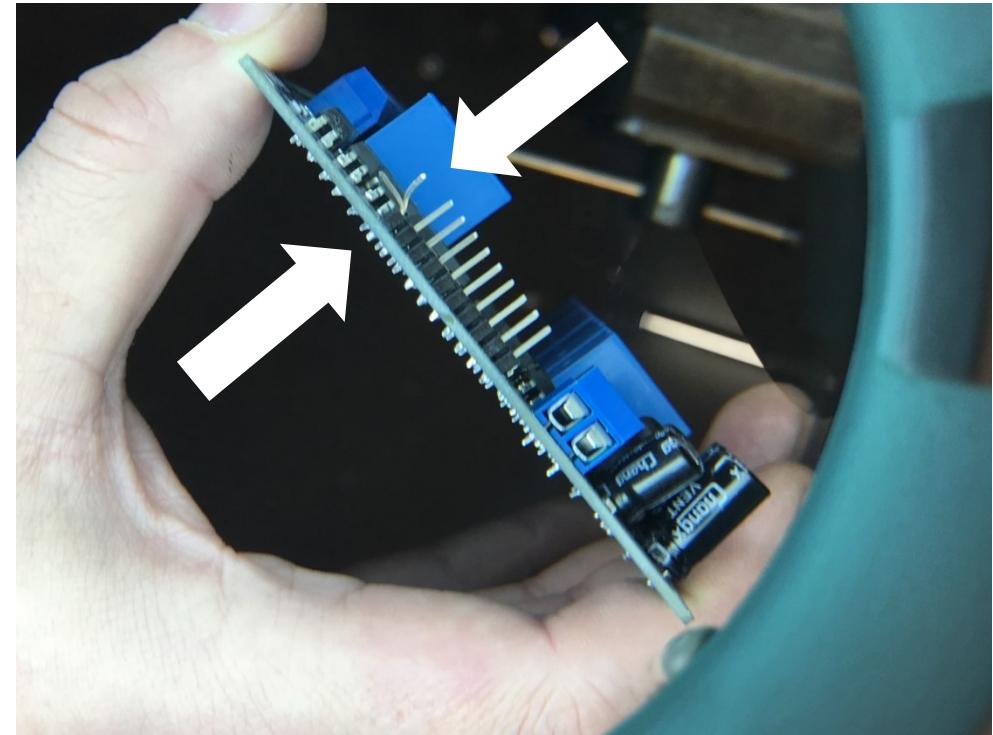
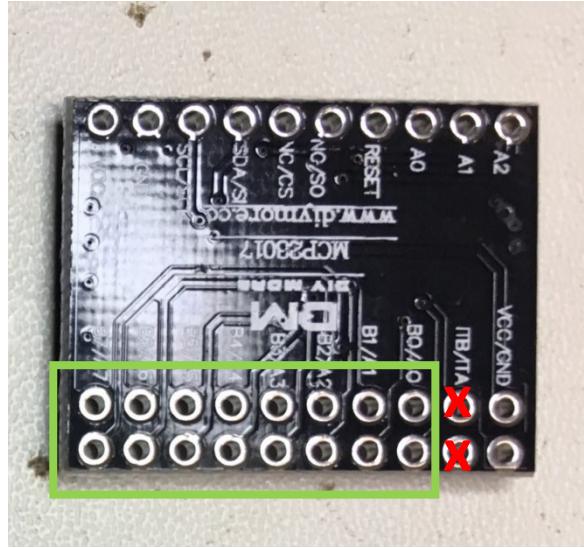
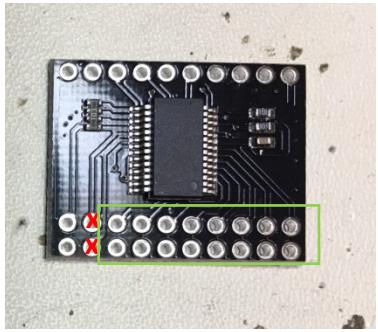
# Das Relaysboard verkabeln... 1

Mit einer flachen Zange müssen die Pins zurecht gebogen werden, damit das MCP-Board auf den Header an der richtigen Stelle sitzt.

2 Pins (3,4) am MCP-Board müssen dabei frei bleiben... Ein nicht ganz leichtes Unterfangen...

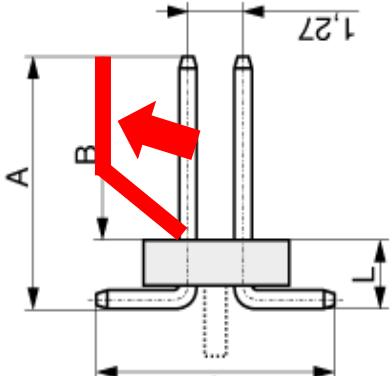


# Das Relaysboard verkabeln... 2

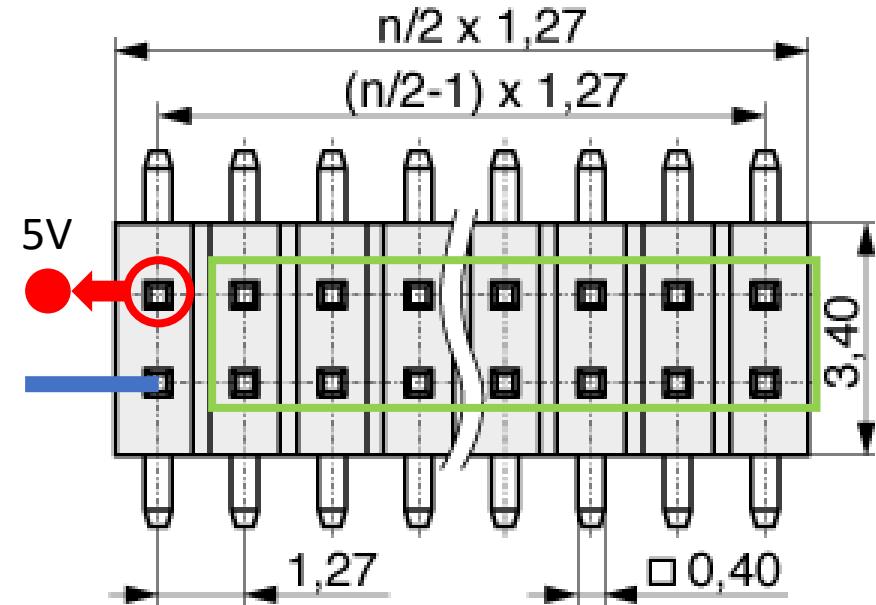
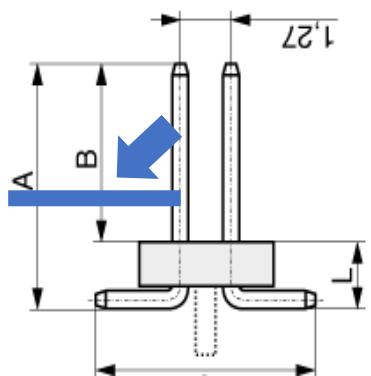


# Das Relaysboard verkabeln... 3

- Pin 1: „Biegen“ um  
1-Raster nach links!



- Pin 2:  
Wird ganz umgebogen!



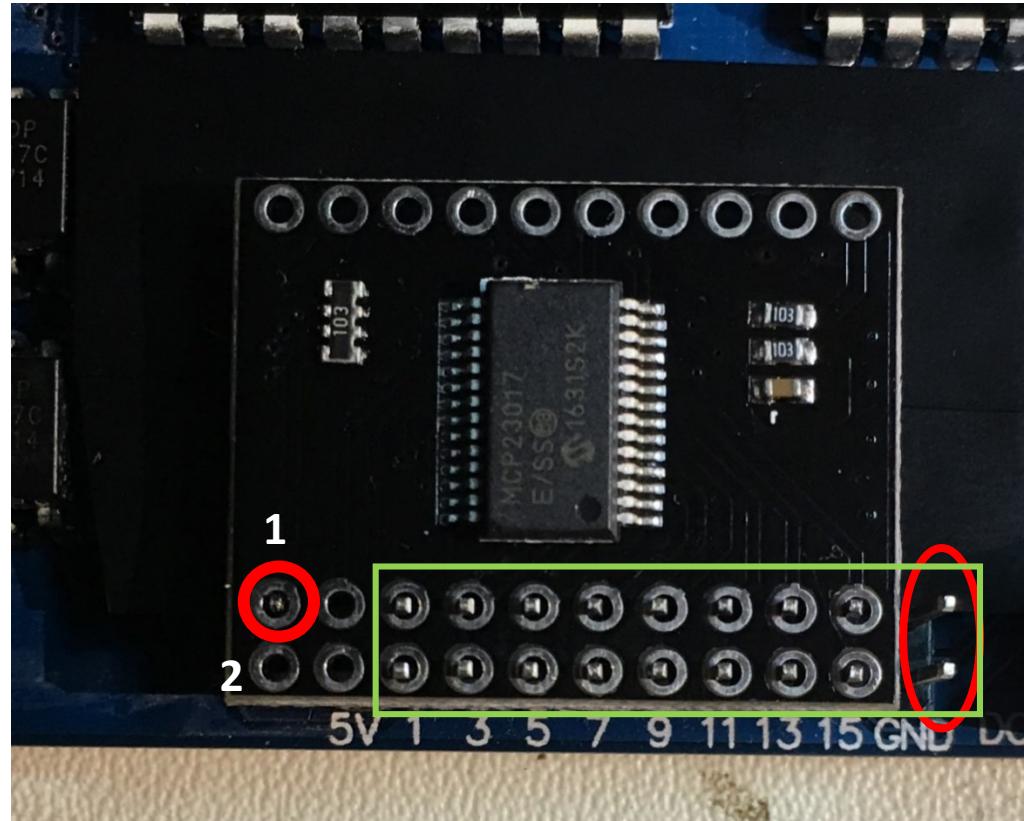
(( (( )) ))

# Das Relayboard verkabeln... 4

Nun wird das MCP-Board auf die vorbereiteten Headerpins gesteckt:

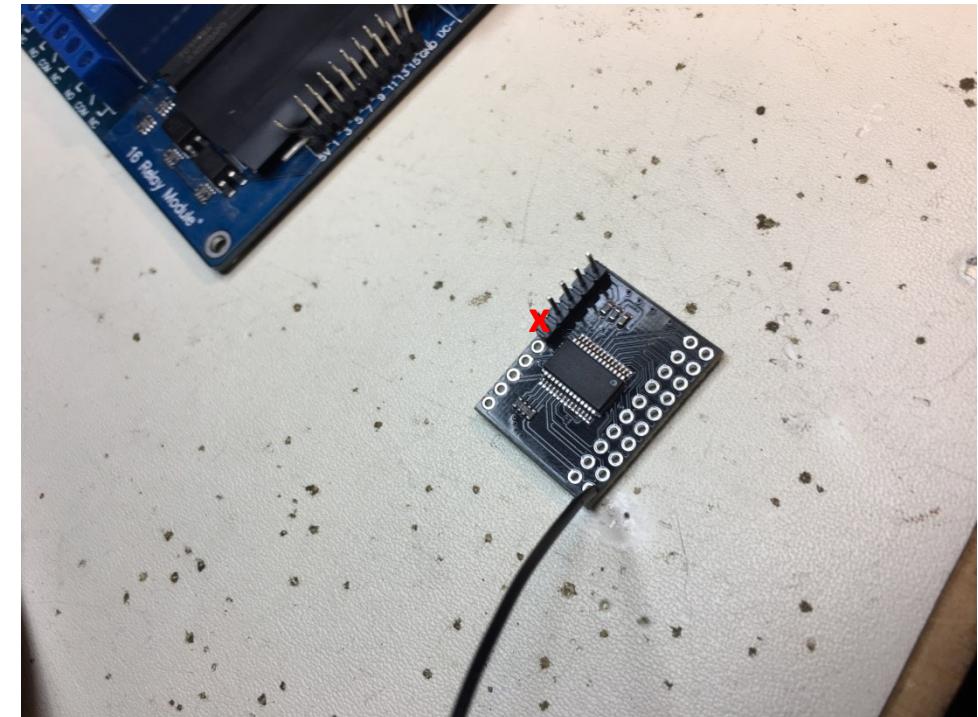
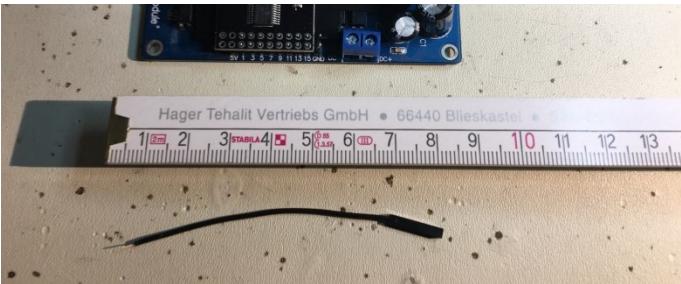
- In -1- sieht man den zum „S“ gebogenen Pin (wird später gelötet).
- Bei -2- ist darunter der komplett umgebogene Pin (nicht zu sehen)
- Stelle mit Pins ¾ ist frei.
- Ganz rechts: Freie GND-Pins (siehe später)

WICHTIG: Noch nicht festlöten!



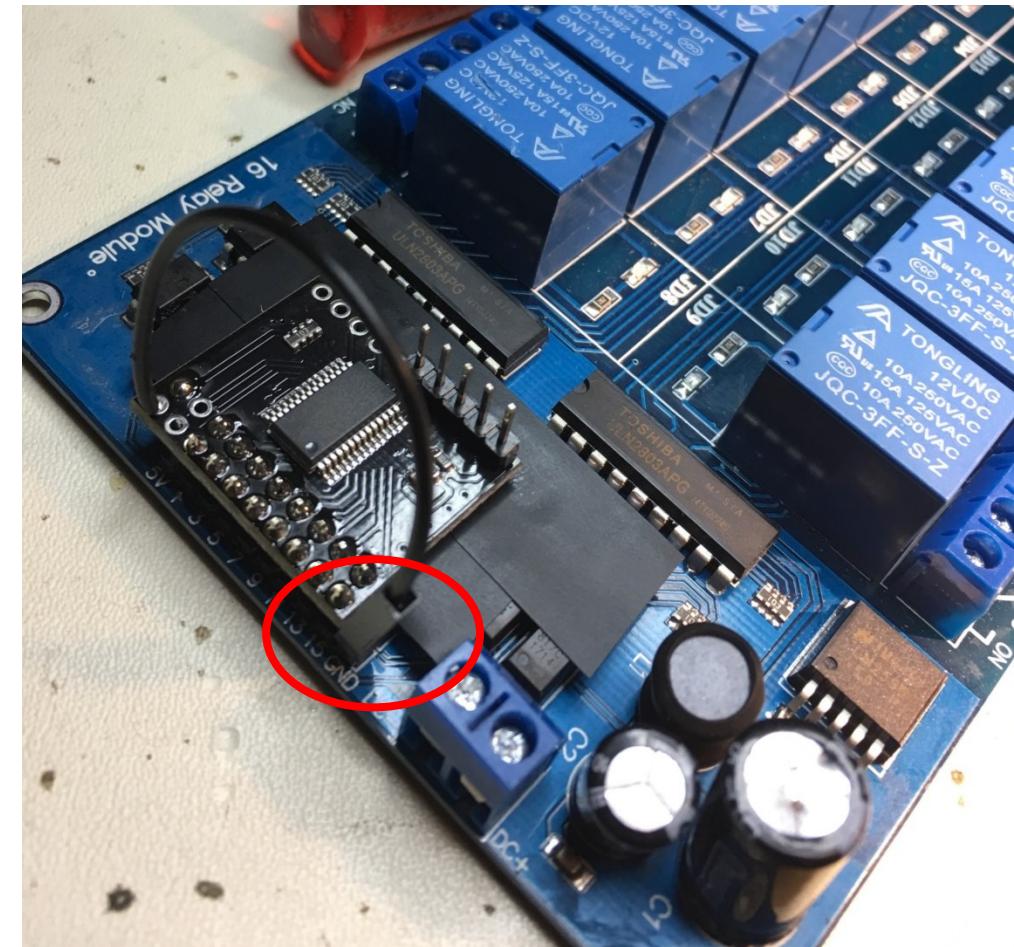
# Das MCP-23017 verbinden

4 Pin-Header (4 reichen :P) und 1x GND-Wire schwarz verlöten:



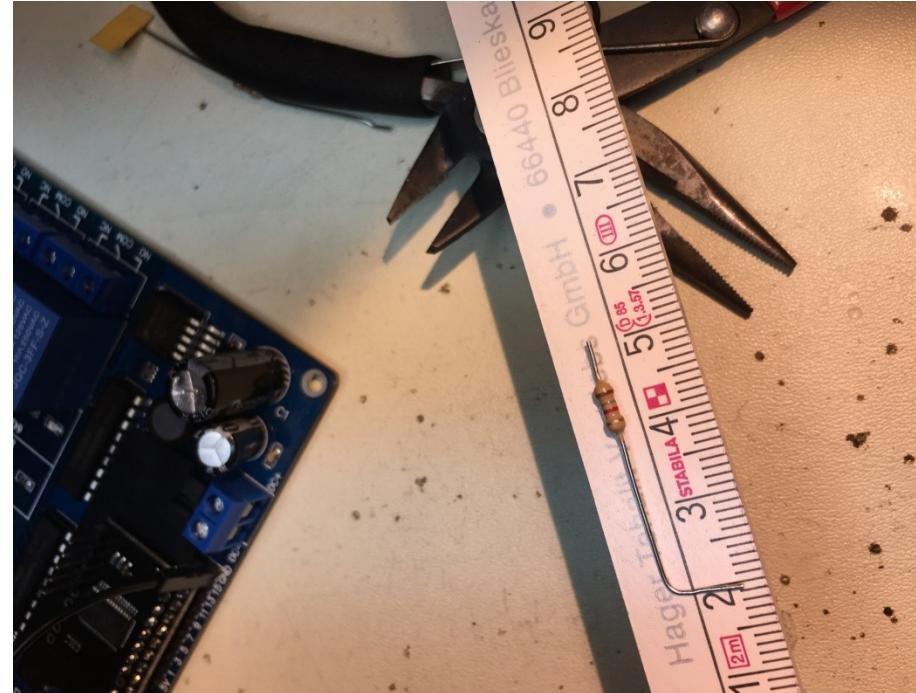
# MCP-23017 auf das RelayBoard löten

- Voraussichtl. Auflagefläche mit Isolierband bekleben
- MCP-Board richtig platzieren
- MCP-PCB mit Relayboard verlöten
- Schwarzen GND-Wire mit RelayBoard verbinden



# Den ESP-12F bestücken 1

- Female Header anlöten & 1. 1k8/2k2 Widerstand vorbereiten:



(((( ))))

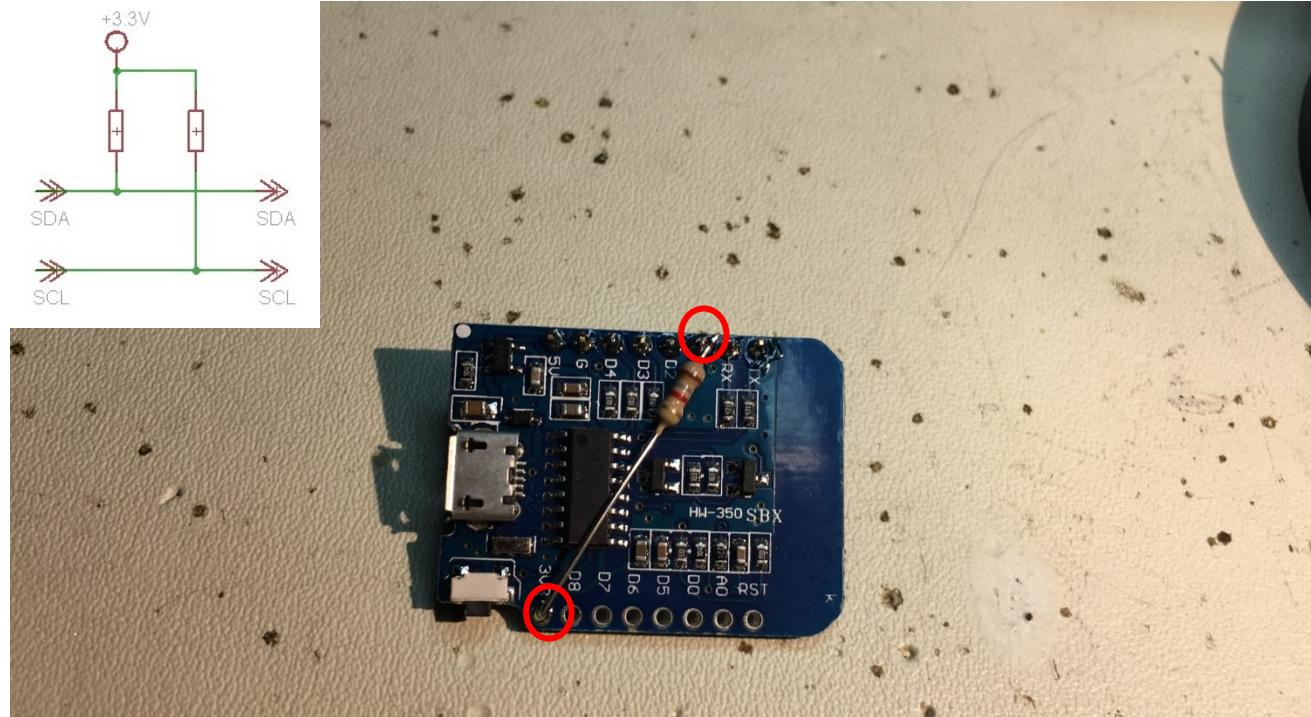
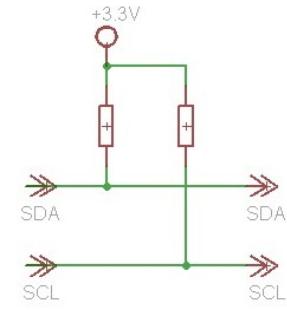


# Den ESP-12F bestücken 2

- 1. 1k8 oder 2k2  
Widerstand anlöten:

Pin D1 → 3V3!

- Achtung: Draht darf keine anderen Kontakte berühren...

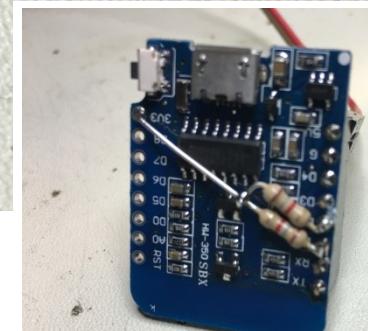
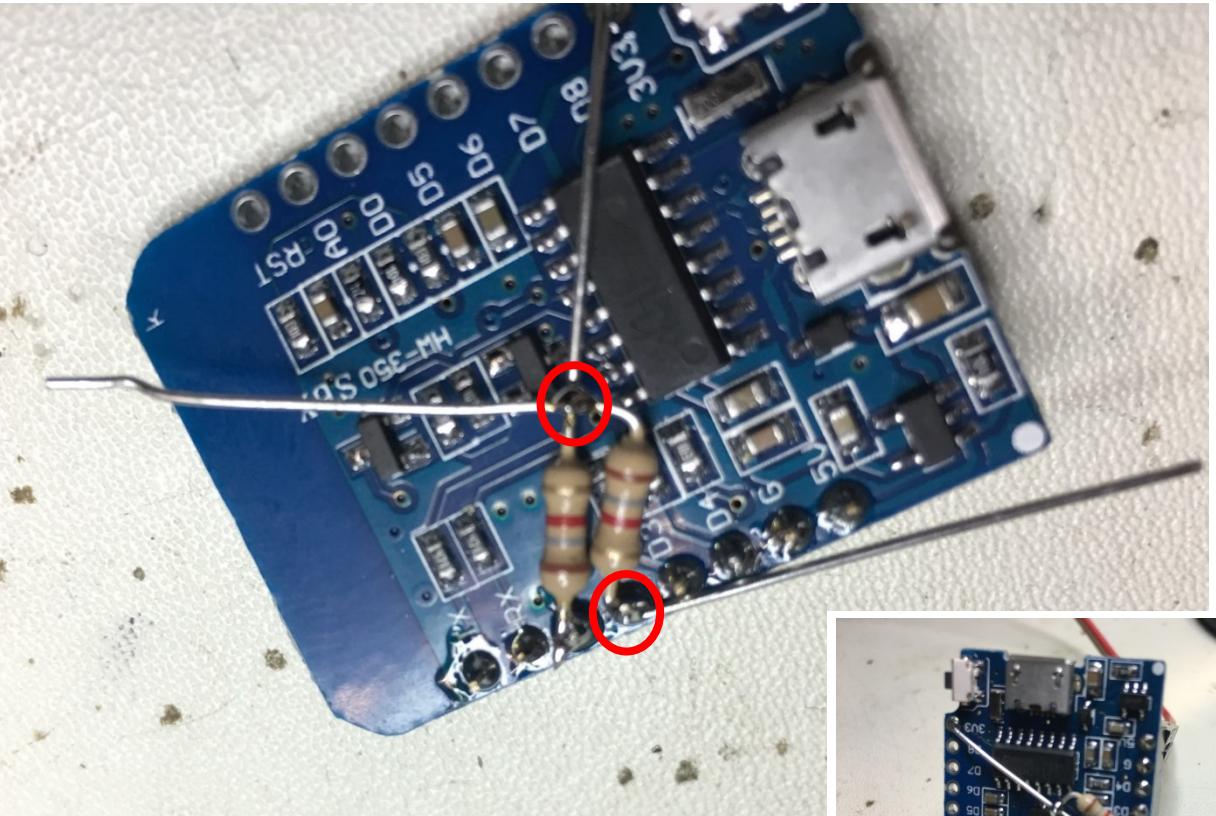


# Den ESP-12F bestücken 3

- 2. 1k8 oder 2k2 Widerstand zu einem „S“ – biegen und verlöten.

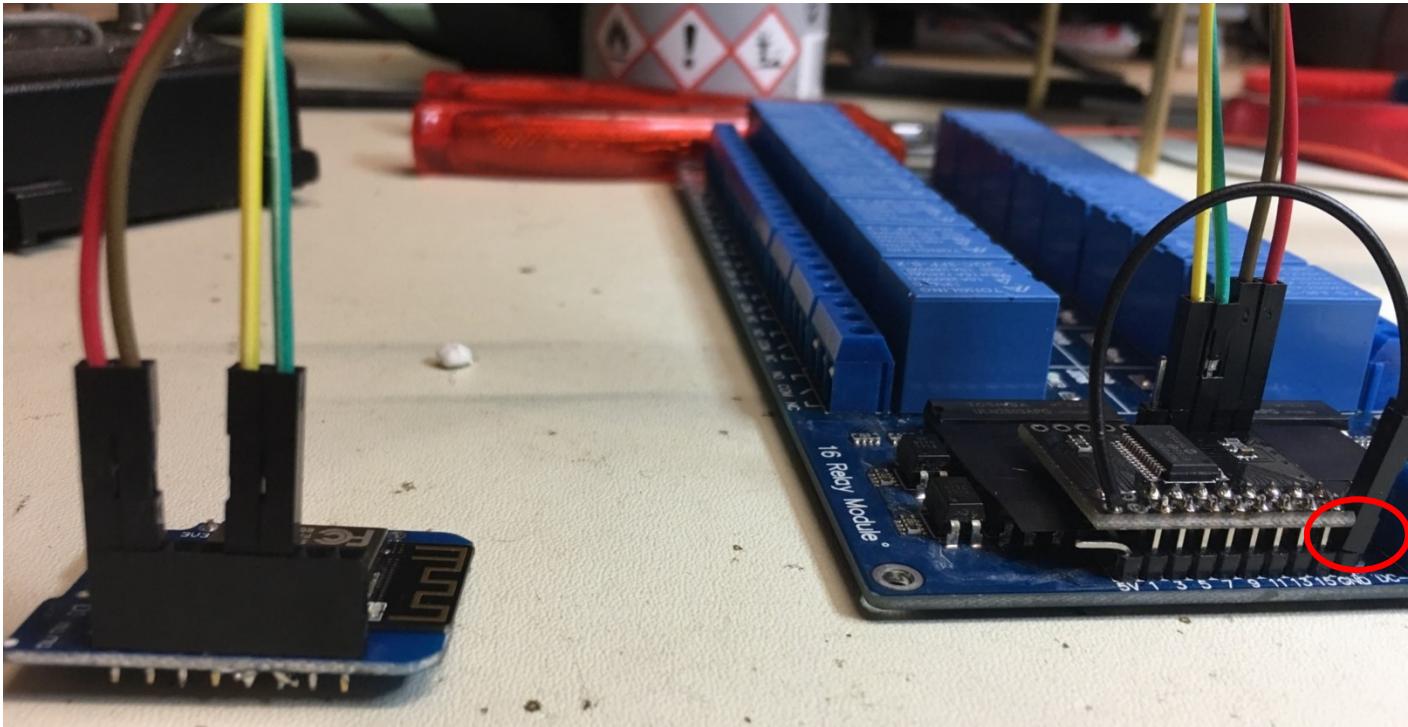
D2 → 1.Widerstand 3V3

- Abschneiden.
- Achtung: Draht darf keine anderen Kontakte berühren...



# Alle drei zusammen...

- Am Wemos von links:  
5V|GND|leer|leer|GELB|GRÜN
- Am MCP von links:  
GELB|GRÜN|GND|5V
- SCHWARZ im freien GND (Relayb.)

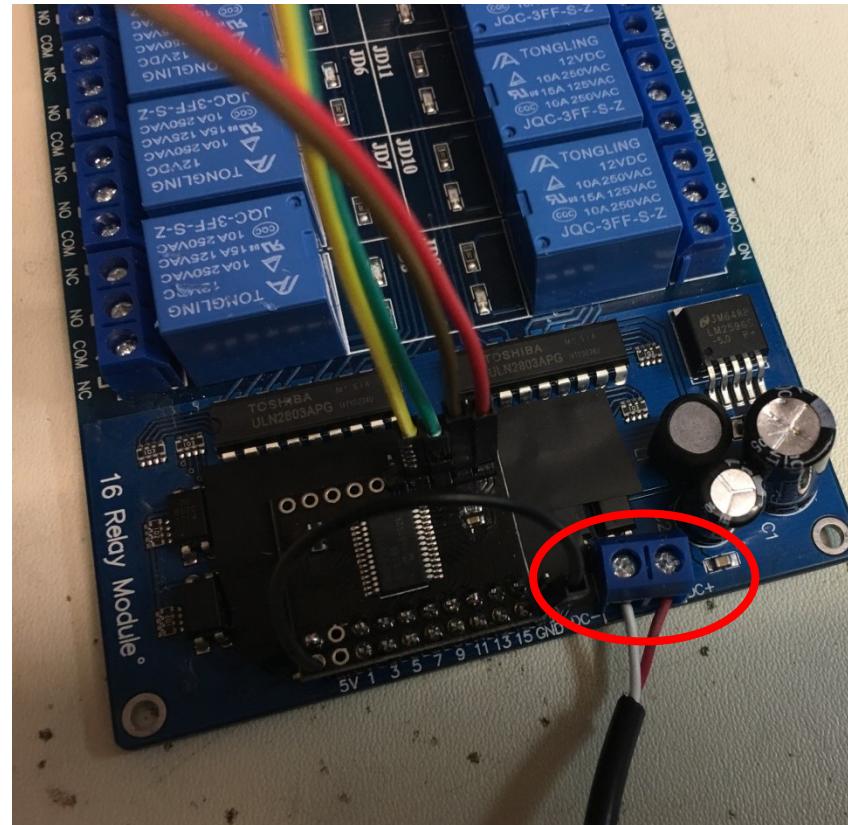
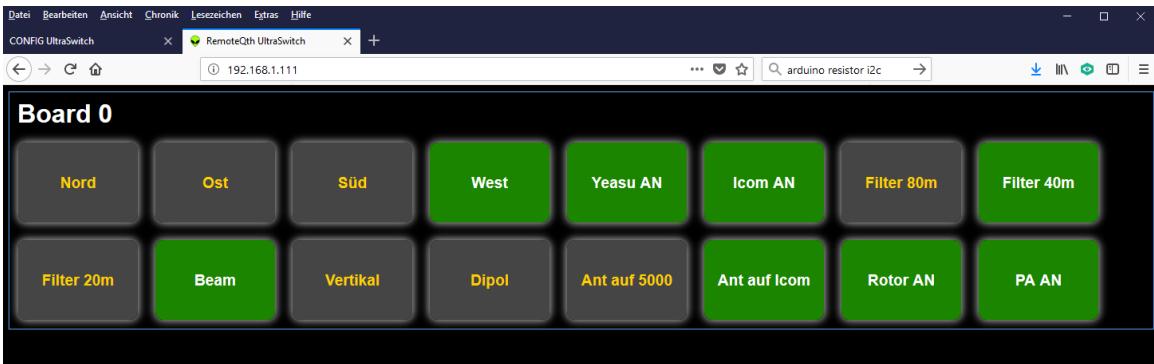


PS: 4 Pins reichen (der 5. war ein Versehen :P)



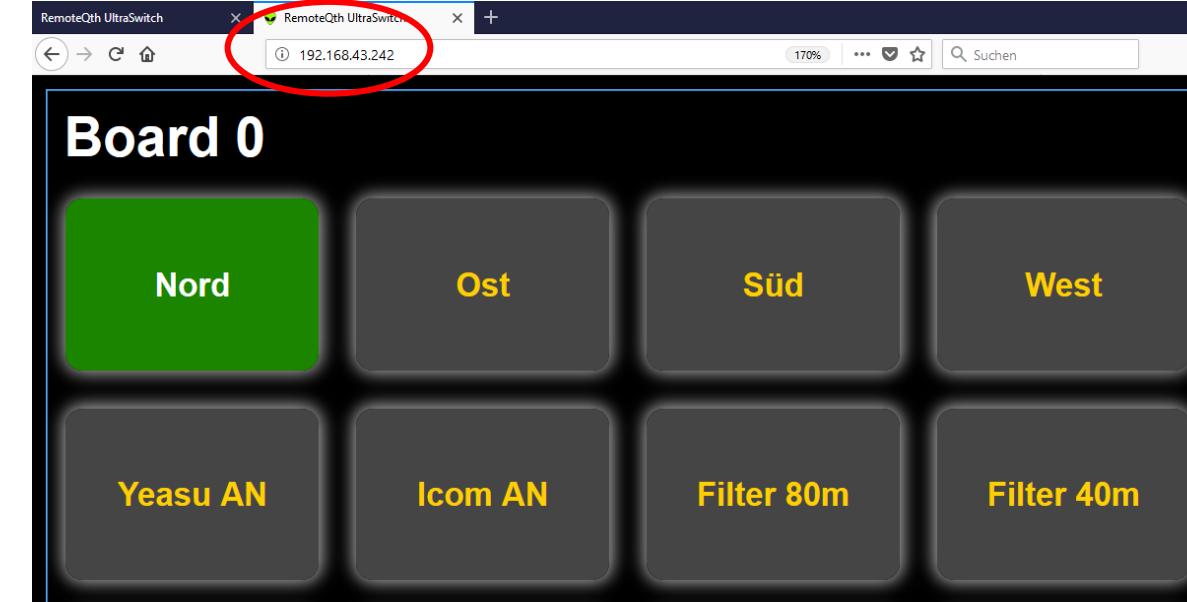
# Am Ende...

- 12V verkabeln und anklemmen. (POLUNG!)
- Netzteil einstecken
- Browser starten



# Zurück zum Anfang... Use Case 1

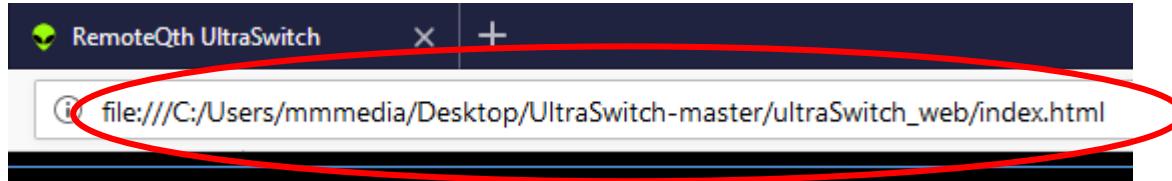
- Der Ultraswitch wird direkt mit seiner IP-Adresse angesprochen.
- Der Ultraswitch antwortet mit seiner eigenen index.html
- Labelbeschriftung, Gruppierungen und Funktionen kann nicht geändert werden und wird von h.mmmedia-online.de geladen und übernommen.



=> Keine Konfiguration notwendig, aber auch nicht möglich. Aber wenn die Anforderungen damit erfüllt werden: **FERTIG!**



# Zurück zum Anfang... UseCase 2... 1

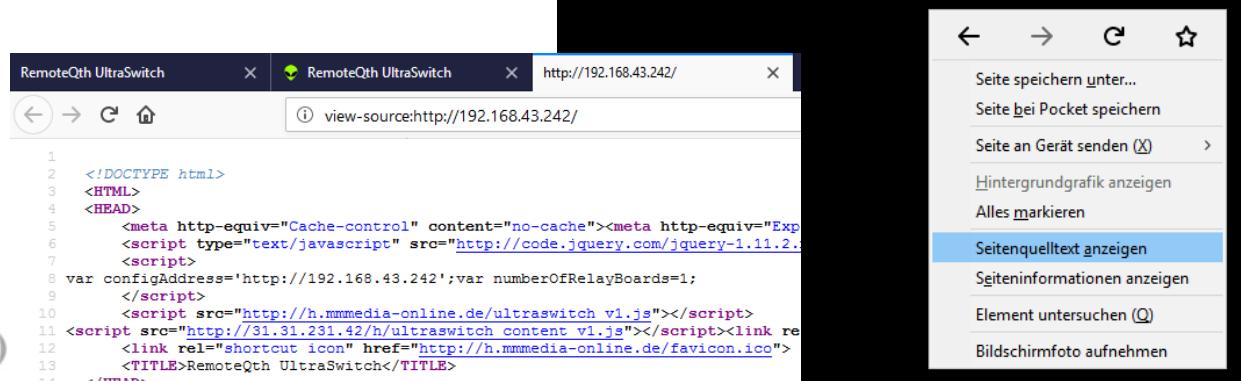
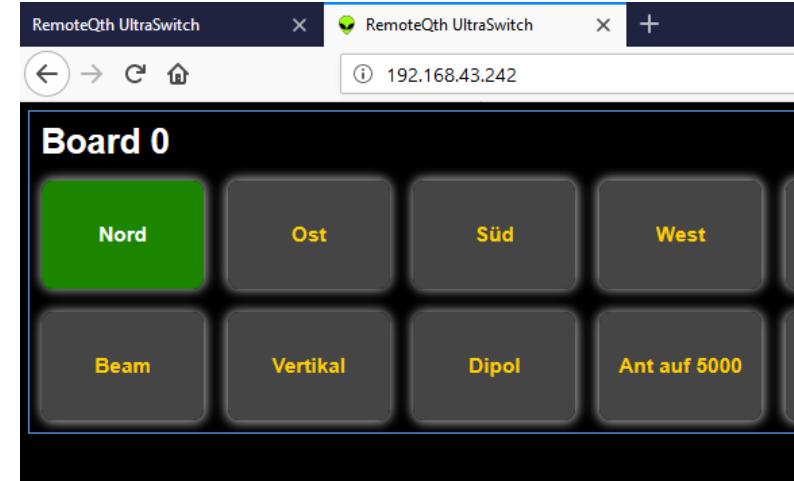


- Der Ultraswitch wird per index.html aufgerufen, welche lokal z.B. auf dem Desktop liegt.
- z.B. [file:///C:/Users/mmma.../ultraSwitch\\_web/index.html](file:///C:/Users/mmma.../ultraSwitch_web/index.html)
- Alle Files Lokal: Eigen Anpassungen möglich!



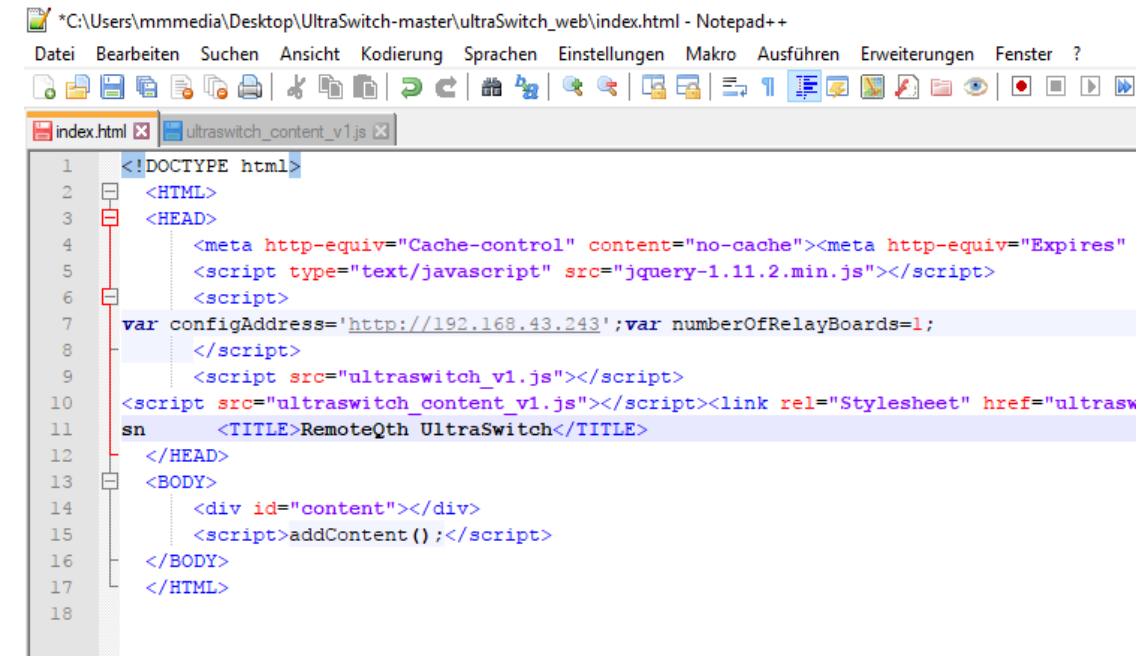
# UseCase 2... Eigene index.html erstellen... 1

- Ultraswitch per IP z.B.  
192.168.43.242 aufrufen.
- Auf den leeren schwarzen  
Hintergrund mit rechter Maustaste =>  
Seitenquelltext anzeigen
- Kompletten Text markieren und in die  
Zwischenablage kopieren.



# UseCase 2... Eigene index.html erstellen... 2

- Die index.html aus dem Ordner ultraswitch\_web mit einem Editor öffnen.
- Text markieren und entfernen
- Vorher kopierten Quellcode einfügen und abspeichern.
- Mit Doppelklick auf diese index.html die Datei im Browser öffnen. Es muss alles wie gewohnt angezeigt werden. Wenn nicht:  
-siehe Folie zu CORS-



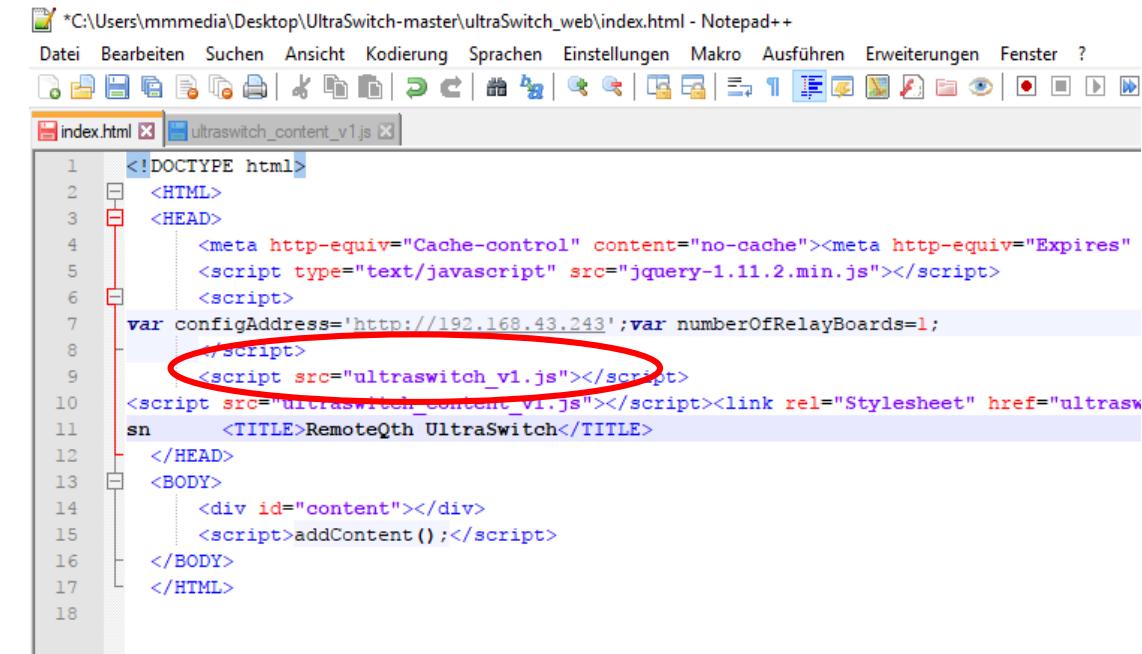
The screenshot shows the Notepad++ editor window with the file 'index.html' open. The code is as follows:

```
<!DOCTYPE html>
<HTML>
<HEAD>
<meta http-equiv="Cache-control" content="no-cache"><meta http-equiv="Expires" content="0">
<script type="text/javascript" src="jquery-1.11.2.min.js"></script>
<script>
var configAddress='http://192.168.43.243';var numberOfRelayBoards=1;
</script>
<script src="ultraswitch_v1.js"></script>
<script src="ultraswitch_content_v1.js"></script><link rel="Stylesheet" href="ultraswitch_content_v1.css" type="text/css" />
<TITLE>RemoteQth UltraSwitch</TITLE>
</HEAD>
<BODY>
<div id="content"></div>
<script>addContent();</script>
</BODY>
</HTML>
```



# UseCase 2... Alles lokal!

- Wenn alle Dateien LOKAL IM GLEICHEN VERZEICHNIS wie die index.html liegen, dann können die Pfadangaben gekürzt werden:  
aus z.B. src=„[http://h.mmmmedia-online.de/ultraswitch\\_v1.js](http://h.mmmmedia-online.de/ultraswitch_v1.js)“ wird  
src=„ultraswitch\_v1.js“
- Wenn alle http://<url>/<filename> in http://<filename> geändert wurden, arbeitet der die Steuerung OHNE Internet
- Problem: jquery muss ebenfalls lokal vorliegen!
- Volle Freiheit für Anpassungen: Content!



```
<!DOCTYPE html>
<HTML>
<HEAD>
<meta http-equiv="Cache-control" content="no-cache"><meta http-equiv="Expires" content="0">
<script type="text/javascript" src="jquery-1.11.2.min.js"></script>
<script>
var configAddress='http://192.168.43.243';var numberofRelayBoards=1;
</script>
<script src="ultraswitch_v1.js"></script>
<script src="ultraswitch_content_v1.js"></script><link rel="Stylesheet" href="ultraswitch_content_v1.css">
<TITLE>RemoteQth UltraSwitch</TITLE>
</HEAD>
<BODY>
<div id="content"></div>
<script>addContent();</script>
</BODY>
</HTML>
```



# Customizing => Labels, Gruppierung, RelaysPerBoard

Einstellungsmöglichkeiten in der **ultraswitch\_content\_v1.js**:

- Labels => textboard0 = Labels für das 1. Board usw...

```
var textboard0 = ["Nord","Ost","Süd","West","Yerasu AN","Icom AN","Filter 80m","Filter 40m","Filter  
20m","Beam","Vertikal","Dipol","Ant auf 5000","Ant auf Icom","Rotor AN","PA AN"];      → Fasse dich kurz!!!
```

- Gruppen => board0Group = Gruppendefinition für das 1. Board usw...

```
var board0Group = [[0,1,2,3],[6,7,8],[9,10,11],[12,13]];
```

Erklärung: Auf dem 1. Board Nummer 0 gibt es 4 Gruppierungen. In der 1. Gruppierung ist Knopf0,1,2,3 miteinander verbunden. Es kann nur immer einer in der Gruppe "AN" sein. Usw...

- **numberOfRelaysPerBoard** = Wie viele Relays pro Board (1., 2., 3.,...)

```
var numberOfRelaysPerBoard = [14,16,16,16,16,16,16,16,16,16,16,16,16,16,16,16];
```

Erklärung: Erstes Board hat 14 Relays, 2. 16, ....



# UseCase 3... Eigener Webserver

Eigener Webserver im Internet oder im eigenen Netzwerke (z.B. RaspiPI):

- Wurde in der Config.html für Content die IP-Adresse zum **EIGENEN WEBSERVER** im Internet angegeben, wird automatisch die IP zur **ultraswitch\_content\_v1.js** und **ultraswitch\_v1.css** generiert:  
Namens-Konvention: `http://<eigene_ip>/h/<dateiname>` erzeugt!

## HowTo:

- Unterverzeichnis mit dem Namen „h“ erstellen im Rootverzeichnis anlegen.
- (Eigene,angepasste) **Ultraswitch\_content\_v1.js** und **ultraswitch\_v1.css** dorthin kopieren.
- **Ultraswitch ganz normal per IP aufrufen! Starten per lokale index.html NICHT NOTWENDIG!**
- Hinweis: Kommt ein eigener [Raspberry PI](#) als Webserver zum Einsatz, einfach dessen IP-Adresse dort verwenden.



# Umschiffen von Engstellen... Wenns mal hängt!

- Aufgrund seines „Alters“ ist die Ethernet-Schnittstelle mit den Optimierungen heutiger Web-Browser überfordert.
- Die Ethernet-Lib kann nur 4 Verbindungen gleichzeitig handeln, aber moderne Browser öffnen Dutzende.
- Irgendwann komm der Mikrocontroller ins straucheln und hängt sich auf.
- Workaround: MicroFox (siehe Verzeichnis „Software“) – eine angepasste Version des Firefox, bei den die Max-Connection-Zahl auf 2 beschränkt wurde. Zwar surft es sich damit nicht mehr gut im Netz, aber der Ultraswitch kann prima damit gesteuert werden. Sollte es also zu Problemen kommen, bitte nur den MicroFox als Browser verwenden. (Auch mobile Firefox können angepasst werden – see about:config)



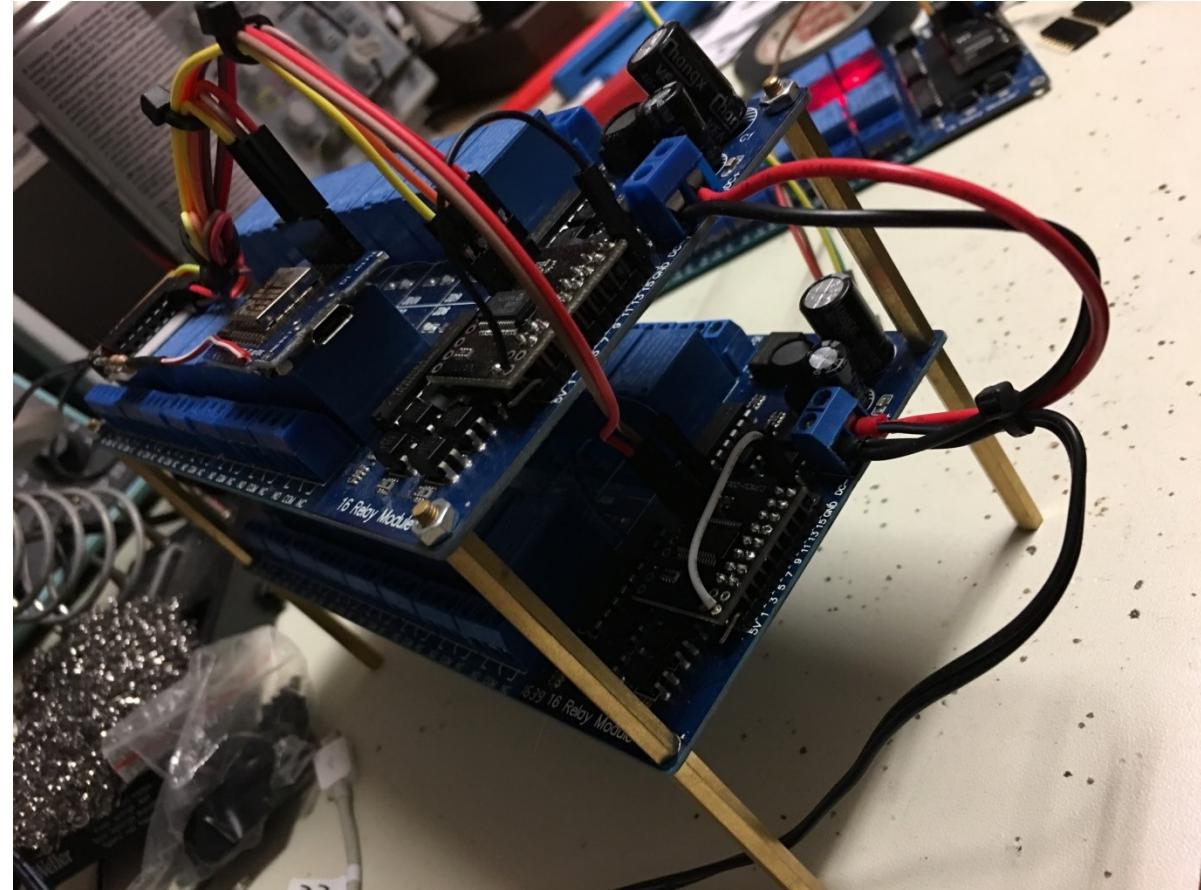
# Aufbau und Funktionsweise des Sourcecodes

- Main-Sketch auf dem ESP12F für Hardwarekontrolle, JSON, WebServer:  
UltraSwitch.ino
- Benutzte Bibliotheken:  
Adafruit-MCP23017 → I2C-Kontrolle MCP23017  
ArduinoJson → JSON Format für das Webserver-Response  
WiFiManager → Einfach Konfiguration des Webservers
- Ultraswitch\_content\_v1.js → Konfiguration der Labels, Gruppen, etc. (*Beschriftungen*)  
Ultraswitch\_v1.js → Schaltlogik und Steuerung der UI-Element  
Ultraswitch\_v1.css → UI Look&Feel, Buttongröße, Farben, etc. (*Buttons*)
- Setup\_esp.html → Vorkonfiguration des ESP12F Access-Points  
Config.html → Konfiguration der IP- und Servereigenschaften  
index.html → Lokale ESP12F Anwendung (Einstiegspunkt)



# Weitere Module...

- Weitere RelayBoards wie beschrieben zusammenbauen
- I2C-Adresse vergeben!  
(nächste Folie)
- Nur Gelb|Grün mit Gelb|Grün verbinden (Parallel-Bus)
- Stromversorgung einfügen
- Config.html → Board-Anzahl
- Fertig!

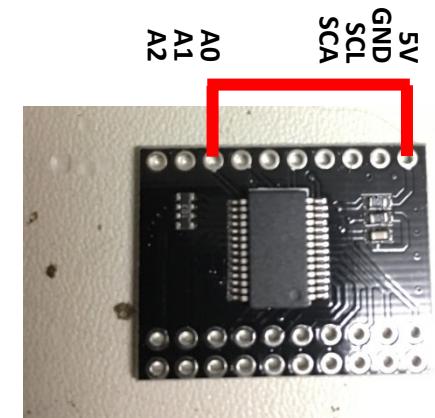
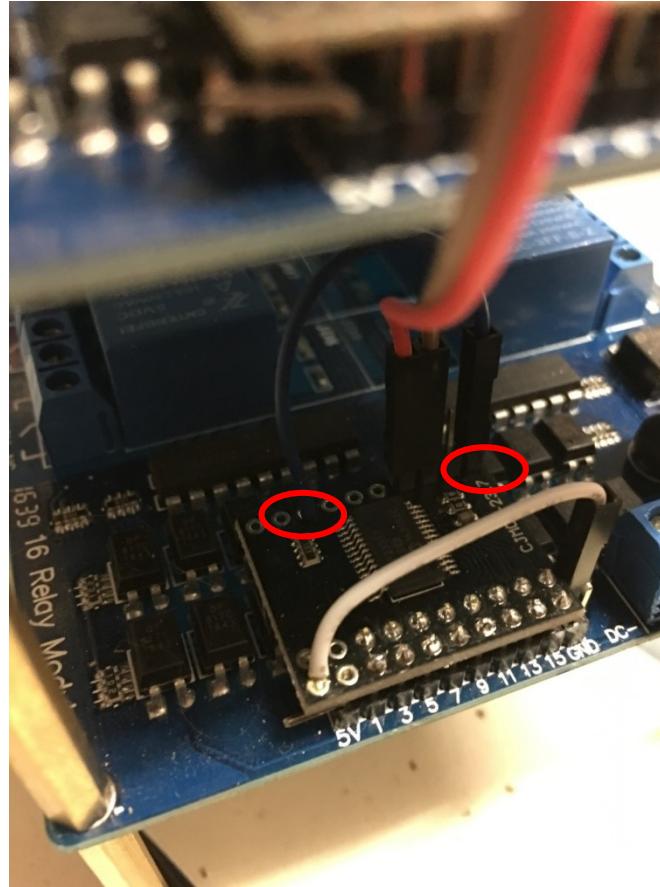


# Weitere Module... I2C Adressierung => Bis zu 8 Devices!

- Die Verbindung mit 5V (HIGH) muss genau in der Reihenfolge durchgeführt werden:  
0 => 1. Board, 1 => 2. Board,  
2=> 3. Board, ...!!!

```
addr 0 = A2 low , A1 low , A0 low  000
addr 1 = A2 low , A1 low , A0 high 001
addr 2 = A2 low , A1 high , A0 low  010
addr 3 = A2 low , A1 high , A0 high 011
addr 4 = A2 high , A1 low , A0 low  100
addr 5 = A2 high , A1 low , A0 high 101
addr 6 = A2 high , A1 high , A0 low  110
addr 7 = A2 high, A1 high, A0 high 111
```

Hinweis: Nicht verbunden => LOW



# Viel Platz für eigene Ideen...

- Mehrere Ultraswitch-Cluster
- Nutzung des ADC-IN für Messungen (Temperatur, etc.)
- Verwendung als Antennenschalter (Wichtig: Abschirmung, Gehäuse...)
- Zeitsteuerung von Pins (siehe mega-webswitch)
- Integration in N1MM (M. Hammelmann, DK5AX), Linuxanwendung
- Remote-Schalter für Remote Station (VPN wird empfohlen)
- Integration mit einem Raspberry PI als Server (UDP, Broadcast)
- ...



# THE END

- Danke für Eure Aufmerksamkeit –

Die verwendete Soft- und Hardware, Architecture und Firmware unterliegt der  
[Attribution-NonCommercial-ShareAlike 4.0 International \(CC BY-NC-SA 4.0\)](#)

Weitere Projekte: [www.remoteqth.com](http://www.remoteqth.com)

Fertige Kits kaufen? Weitere Fragen? Email: [dm5xx@gmx.de](mailto:dm5xx@gmx.de) -

